UNIVERSIDADE DO MINDELO
DEPARTAMENTO DE ENGENHARIA E RECURSOS DO MAR

CURSO DE LICENCIATURA EM INFORMÁTICA DE GESTÃO

RELATÓRIO DE PROJETO DE LICENCIATURA

ANO LETIVO 2014/2015 – 4º ANO

Autor: Frederico Silva Soares, N.º 2726

Mindelo, 2015

UNIVERSIDADE DO MINDELO

Sapientia Ars Vivendi

Frederico Silva Soares

UMTÁXI - APLICAÇÃO MÓVEL - LOCALIZAÇÃO E REQUISIÇÃO DE TÁXI

Trabalho de Conclusão de Curso apresentado à **Universidade do Mindelo** como parte dos requisitos para obtenção do grau de licenciatura em Informática de gestão

Orientador:
Doutor João Dias

Mindelo, 2015

Dedico este trabalho a todos que de uma forma ou de outro me ajudaram durante estes anos de formação, em especial a minha família, os meus professores, os meus colegas e amigos.

AGRADECIMENTOS

A vida é feita de desafios, assumir e ultrapassar e este foi mais um com todo o meu esforço e dedicação juntamente com as pessoas que me rodeiam. Cumprir e realizar um sonho pessoal adiado nunca é tarde para correr atrás dele.

Começo por agradecer a Deus e aos meus pais Lucília Soares e José Soares, os quais incentivaram-me e apoiaram-me durante todo o tempo, ajudando-me de todas as formas, dividindo as tristezas, angustias, sem contar os momentos de mau humor e de teimosia, mas também dividiram excelentes momentos de alegria, de vitórias e sempre apoiando para que eu crescesse cada dia mais, não esquecendo também da ajuda financeira disponibilizada por eles no decorrer do curso, a eles devo o meu mais sincero e profundo amor.

Também agradeço a minha tia Maria do Carmo Benrós e o seu marido Carlos Benrós, que disponibilizaram-me a casa deles para que eu hospedasse durante todo o meu percurso de formação.

Á minha avó de coração Ana Júlia agradeço por todo o carinho, incentivo e concelhos que ela deu-me durante todo tempo.

Finalmente agradeço aos meus colegas, docentes e amigos que estiveram ao meu lado desde o início da minha jornada, os quais fizeram grande diferença na minha vida, e que nunca serão esquecidos.

A todos as pessoas que direta ou indiretamente contribuíram para o meu sucesso durante este percurso, um sincero e de coração muito obrigado!

"*Que os vossos esforços desafiem as possibilidades, lembrai-vos de que as grandes coisas do homem foram conquistadas do que parecia impossível.* "

Charles Chaplin

RESUMO

O mercado de dispositivos móveis vem crescendo cada vez mais, e com isso as funcionalidades dos aparelhos se tornam essenciais na hora da escolha dos mesmos. Nesse sentido o desenvolvimento de aplicativos para dispositivos móveis vem crescendo consideravelmente. Com isso a busca por uma plataforma moderna e ágil para o desenvolvimento de aplicações se torna de extrema importância. Levando em conta todos esses fatores, o presente trabalho tem como objetivo desenvolver um aplicativo baseado em localização de táxis chamado "UM Táxi". Tal aplicativo possibilita ao utilizador visualizar os taxis mais próximo dele num mapa e possibilita a requisição do taxi pretendido através de uma chamada telefônica ou envio de uma mensagem escrita, desde que o mesmo aceite o serviço, ao usuário final. O aplicativo foi desenvolvido para plataforma Android, e um WebService em PHP com acesso ao banco de dados MySQL.

Palavras-chave: Android, Sistema Posicionamento Global, Computação Móvel, WebService, Localização de pontos.

ABSTRACT

The mobile device market is growing more and more, and thus the functionality of the devices become essential when choosing them. In this sense, the development of applications for mobile devices is growing considerably. Thus, the search for a modern and agile platform for application development becomes of utmost importance. Taking into account all these factors, this study aims to develop an application based on location taxis called "A Taxi". Such application enables the user to view the nearest taxis it on a map and allows you to request the intended taxi through a phone call or sending a text message, provided it accepted the service to the end user. The application was developed for Android platform, and a Web Service in PHP to access the MySQL database.

Keywords: Android, Global Positioning System, Mobile Computing, Web Service, Location points.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

ADT - *Android Development Tools*

ANAC - *Agência Nacional das Comunicações*

API - *Application Programming Interface*

APP – *Aplicativo*

GPS - *Global Positioning System*

HTML - *Hypertext Markup Language*

HTTP - *HyperText Transfer Protocolo*

IDC - *International Data Corporation*

IDE - *Integrated Development Environment*

JSON - *JavaScript Object Notation*

NASA - *National Aeronautics and Space Administration*

NAVSTAR - *NAVigation Satellite with Time And Ranging*

OS - *Operational System*

PHP - *Personal Home Page*

PPS - *Precise Positioning Service*

REST - *Representation State Transfer*

SDK - *Software Development Kit*

SGBD - *Sistema de Gerenciamento de Banco de Dados*

SQL - *Structured Query Language*

SMS - *Short Message Service*

SOAP - *Simple Object Access Protocol*

SPS - *Standard Positioning Service*

TCP - *Transmission Control Protocol*

UI - *User interface*

UML - *Unified Modeling Language*

URL - *Uniform Resource Location*

XML - *Extensible Markup Language*

WEB - *World Wide Web*

1. INTRODUÇÃO

1.1. Contextualização

Deste os tempos primórdios que o homem vem procurando formas de melhorar a sua qualidade de vida, e nos dias que correm tem-se notado que os prestadores de serviços moveis estão cada vez mais preocupados em garantir a comodidade dos seus clientes, procurando antecipar as suas necessidades de modo a mantê-los sempre satisfeitos.

Resultados disso há neste momento no mercado uma grande demanda de dispositivos moveis que agregam as mais diversas funcionalidades denominados de telefones inteligentes ou simplesmente de smartphones, e que possui um público cada vez maior.

De acordo com a pesquisa realizada pela IDC (International Data Corporation) o mercado de smartphones em todo o mundo cresceu 23,1% no segundo trimestre de 2014 em relação ao ano anterior, estabelecendo um novo recorde único trimestre de volume de 295,3 milhões. Na sequência de um primeiro trimestre muito forte, o mercado cresceu 2,6% sequencialmente, alimentado pela demanda contínua para computação móvel e uma abundância de smartphones de baixo custo.

A IDC prevê que o mercado irá continuar a aumentar em ritmo acelerado no segundo semestre do ano e superando 300 milhões de unidades pela primeira vez em um único trimestre.

Diante do exposto é de referir que as oportunidades de desenvolvimento de aplicativos moveis que permitem solucionar problemas do dia-a-dia do utilizador tem-se tornado viável devido à grande demanda do mercado, e é com base em todo isso que se viu-se que seria uma mais valia desenvolver um protótipo para dispositivos móveis usando o sistema operacional Android, para localização e requisição de táxis.

1.2. Motivação

O primeiro fator motivador que levou à criação desse projeto é que na atualidade, há uma crescente preocupação com o tempo. As pessoas estão sempre com pressa, prazos e datas são estabelecidos para qualquer tipo de atividade. Vive-se em função de um cronograma.

De encontro a essa demanda por pontualidade, a maioria das pessoas encontram ás vezes a necessidade de requisitar os serviços de táxis mas muitas vezes torna-se um pouco complicado ou mesmo embaraçoso ter que ir até à estrada e ficar à espera que encontrar um táxi que não esteja ocupado e isso acaba por nos atrasar ainda mais, e mesmo que uma pessoa tenha um número de um taxista e o chama este por sua vez pode estar muito longe e diz quase sempre que está ali perto com o intuito de não perder um frete e a pessoa acaba por esperar um longo tempo e o atraso torna-se ainda maior, imagina-se num dia de chuva que não dá para ir até à estrada procurar um táxi, ou á noite quando saímos de ou para ir a uma festa.

Outro fator motivador foi pelo facto de ter-se notado que o mercado dos dispositivos moveis na atualidade tem-se crescido a um ritmo bastante acelerado por todo mundo, resultados disso pode-se notar que as nossas duas operadoras aqui em Cabo Verde a Unitel T+ e CVMóvel tem-se apostado e muito na venda desses tais dispositivos, o que permite que muitas pessoas possam adquirir uma.

Outro fator é que devido ao crescimento dos dispositivos moveis o mercado para desenvolvimento de aplicações moveis tem-se tornado bastante viável e futuramente poderá ser uma das áreas com maior oportunidade de emprego em relação à tecnologia daí o meu grande interesse em ampliar o meu conhecimento.

Diante de tudo isso, eu e o meu orientador vimos na possiblidade de elaborar algo inovador que pudesse solucionar os problemas do dia-a-dia dos usuários daí a ideia de criar uma aplicação para dispositivos móveis que permite a requisição de taxis permitindo aos utilizadores minimizar tempo de espera.

1.3. Objetivos

1.3.1. OBJETIVOS GERAIS

O projeto tem como objetivo geral desenvolver um protótipo de um aplicativo para *smartphones* com sistema operacional Android que permita ao utilizador através da sua localização utilizando o GPS do dispositivo visualizar os táxis mais próximos e através de uma chamada telefónica ou de um SMS requisitar o táxi pretendido.

1.3.2. OBJETIVOS ESPECÍFICOS

Os objetivos específicos do trabalho são:

a) Implementar o sistema utilizando a plataforma Android;
b) Utilizar banco de dados *MySQL*;
c) Utilizar um WebService desenvolvido em *PHP;*
d) Disponibilizar uma aplicação para dispositivos móveis que permite a requisição de táxis, com associação do GPS do dispositivo;
e) Determinar a posição de um veículo e transmitir suas coordenadas a um servidor remoto;
f) Mostrar a localização de veículos com a integração Google Maps;
g) Calcular a distância entre o passageiro e os taxis
h) Mostrar informações dos taxistas
i) Permitir a requisição e o atendimento dos serviços de taxis através de uma chamada telefônica ou de um SMS.

1.4. Contribuições Esperadas

Espera-se que, com a realização desse projeto, seja possível apresentar um aplicativo de fácil interação e que facilite a vida dos utilizadores, de forma a otimizar o tempo dessas pessoas e evitar possíveis atrasos com os seus compromissos.

1.5. Metodologia

Para a realização deste projeto numa primeira fase foram feitas pesquisas dos dados teóricos através da internet e de bibliografias, na segunda fase procedeu-se com o levantamento dos requisitos do sistema, a modelagem do Sistema, na terceira fase fez-se o estudo da plataforma Android, do API da Google Maps, Web Service, na quarta fase foram instalados e configurados os ambientes de desenvolvimento.

Depois disso na quinta fase deu-se a elaboração do protótipo, a seguir na sexta fase foi feita a implementação do sistema e à elaboração dos testes para uma melhor avalição das funcionalidades do sistema e fazer os reajustes necessários e no fim a conclusão do relatório final.

1.6. Estrutura do trabalho

De acordo com os objetivos definidos, este trabalho é estruturado em quatro capítulos.

Antes da iniciação dos capítulos, temos a contextualização, a motivação, objetivo geral, objetivos específicos, as contribuições esperadas e a metodologia.

No primeiro capítulo, apostou-se por fazer uma abordagem à computação móvel, ao sistema de posicionamento global (GPS), à localização de pontos e da plataforma Android no qual procedeu-se a clarificação dos conceitos básicos que apresentam-se ser pertinentes para trabalho.

No segundo capítulo fez-se a abordagem sobre a arquitetura do sistema, análise dos requisitos e a modelagem do sistema.

No terceiro capítulo fez-se uma abordagem sobre as ferramentas e tecnologia utilizadas no desenvolvimento do projeto e a demonstração alguns trechos de código e de algumas telas do protótipo abordada neste trabalho referente à interação do utilizador com o sistema.

No quarto capítulo fez-se a análise das considerações finais, dando algumas opiniões a respeito do trabalho e a conclusão do mesmo.

2. ABORDAGEM TEÓRICA E DEFINIÇÕES DE CONCEITOS

Neste capítulo serão apresentados conceitos sobre, o sistema de posicionamento global (GPS), a localização de pontos e a computação móvel de uma forma geral bem como da plataforma Android.

2.1. Sistema de posicionamento global (GPS)

De acordo com a National Aeronautics and Space Administration (NASA), o Sistema de Posicionamento Global (GPS), é um sistema de rádio navegação baseado no espaço, que ajuda identificar a posição tridimensional fornecendo em nano-segundo de tempo preciso em qualquer lugar na Terra de cerca de um metro de precisão (por exemplo, latitude, longitude e altitude).

GPS tem suas origens em meados dos anos 1960 quando os cientistas foram capazes de controlar o satélite com mudanças em seu sinal de rádio conhecido como o "Efeito Doppler". Essas experiências de navegação por satélite foram realizadas pela marinha dos Estados Unidos para rastrear submarinos que transportavam mísseis nucleares. Com seis satélites orbitando os polos, os satélites foram capazes de observar as mudanças dos submarinos em Doppler e identificar a localização do submarino em questão de minutos.

No início da década de 1970, o Departamento de Defesa queria garantir um sistema de navegação robusto e estável e estaria disponível a abraçando ideias anteriores de cientistas da Marinha. O Departamento de Defesa então decidiu usar satélites para apoiar o seu sistema de navegação proposto. E em 1978 lançou seu primeiro sistema de navegação NAVSTAR (*NAVigation Satellite with Time And Ranging*). O sistema de 24 satélites tornou-se plenamente operacional em 1993.

Hoje, o GPS é um multiuso, sistema de rádio navegação baseado no espaço de propriedade do Governo dos EUA e operada pela Força Aérea dos Estados Unidos para atender a defesa nacional, segurança interna, civil, comercial, e das necessidades científicas. GPS atualmente oferece dois níveis de serviço: Serviço de

Posicionamento Padrão (SPS- Standard Positioning Service) e Serviço de Posicionamento Preciso (PPS- Precise Positioning Service). O acesso ao PPS é restrito a Forças Armadas dos EUA, agências federais dos EUA, e aliados das forças armadas e do governo.

O SPS está disponível para todos os utilizadores em uma base contínua, mundial, isenta de quaisquer taxas de utilização diretos. Os recursos específicos fornecidos pelo SPS são publicados no posicionamento Padrões de Desempenho do Sistema Globais e especificações.

2.1.1. SEGMENTOS DO SISTEMA GPS

Ainda segunda a NASA, o funcionamento do GPS é suportado por três partes diferentes ou segmentos, o Segmento Espacial, o Segmento de Controle e o Segmento Usuário.

2.1.1.1. SEGMENTO ESPACIAL

De acordo com MONICO (2000), o segmento espacial consiste de 24 satélites distribuídos em seis planos orbitais igualmente espaçados, com quatro satélites em cada plano, numa altitude aproximada de 20.200 km. Os planos orbitais são inclinados 55° em relação ao Equador e o período orbital é de aproximadamente 12 horas siderais. Dessa forma, a posição dos satélites se repete, a cada dia, 4 minutos antes que a do dia anterior. Essa configuração garante que, no mínimo, quatro satélites GPS sejam visíveis em qualquer local da superfície terrestre, a qualquer hora.

As Figuras 1 e 2 ilustram, respectivamente, a constelação dos satélites GPS e a distribuição destes em cada um dos planos orbitais.

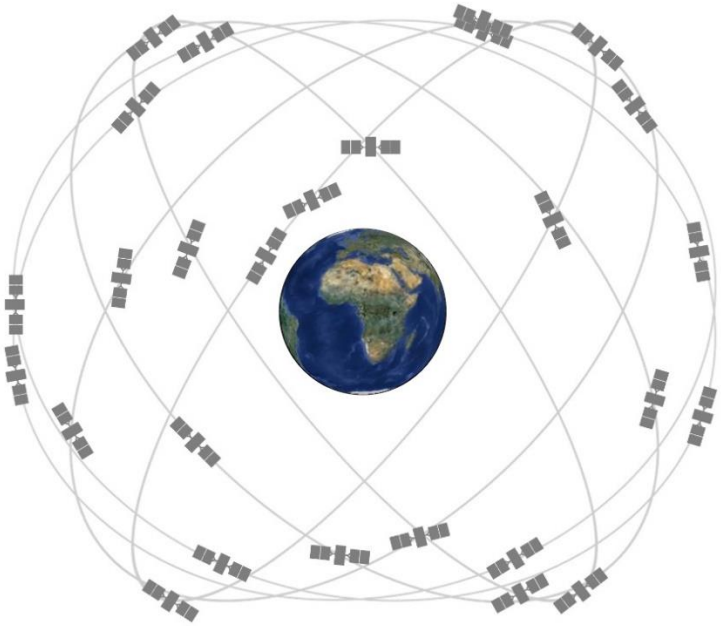

Figura 1: Constelação dos satélites GPS

Fonte: http://www.nasa.gov/directorates/heo/scan/communications/policy/GPS.html

2.1.1.2. SEGMENTO DE CONTROLE

Segundo BERNARDI, J.V.E. & LANDIM, P.M.B., o segmento de controlo tem como principais tarefas monitorar e controlar continuamente o sistema de satélites, determinar o tempo GPS, calcular as correções dos relógios dos satélites, predizer as efemérides dos satélites e atualizar periodicamente as mensagens de navegação de cada satélite.

Esse sistema é composto por cinco estações de monitoramento mundial que estão localizadas nos seguintes locais: Hawai (EUA), Atol Kwajalein (Oceano Pacífico Norte), Ilha de Ascension (Oceano Atlântico Sul), Ilha de Diego Garcia (Oceano Indico Sul) e Colorado Springs (EUA); três delas com antenas para transmitir os dados para os satélites (Ilha Ascension, Ilha de Diego Garcia e Atol de Kwajalein); e uma estação de controle central (Master Control Station) localizada em Colorado Springs.

A figura a seguir ilustra a estação de controlo e monitoramento de GPS.

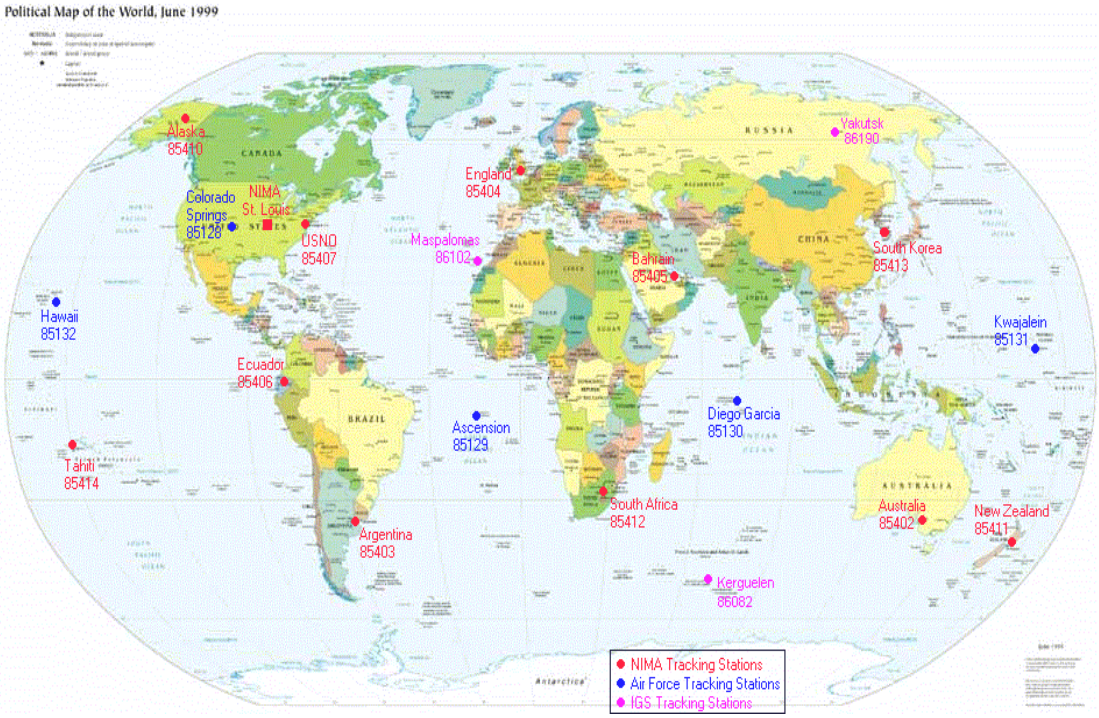

Figura 3: Estação de controlo e monitoramento de GPS

Fonte: BERNARDI, J.V.E. & LANDIM, P.M.B

2.1.1.3. SEGMENTO DE UTILIZADOR

De acordo com MONICO (2000), o segmento de utilizador é constituído pelos receptores GPS, os quais devem ser apropriados para os propósitos a que se destinam, tal como em navegação, Geodésia ou outra atividade qualquer. A categoria de usuários pode ser dividida em civil e militar.

Os militares fazem uso dos receptores GPS para estimar suas posições e deslocamentos quando realizam manobras de combate e de treinamento. Atualmente, há uma grande quantidade de receptores no mercado civil, para as mais diversas aplicações, limitadas apenas pela imaginação dos usuários, o que demonstra que o GPS realmente atingiu sua maturidade.

A figura a seguir ilustra o utilizador utilizando os receptores GPS.

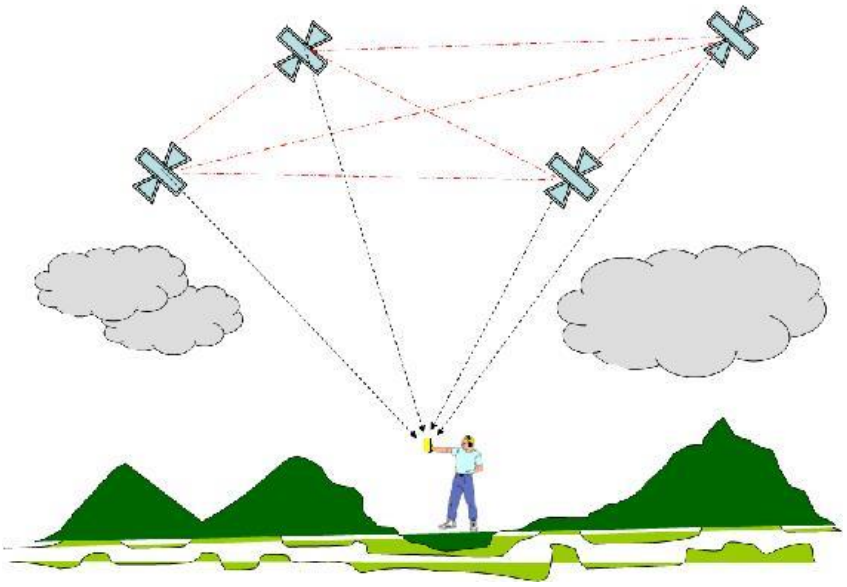

Figura 2: Usuário utilizando os receptores GPS

Fonte: http://www.vaztolentino.com.br/conteudo/562-TECNOLOGIA-GPS-SEGMENTO-ESPACIAL-SEGMENTO-DE-CONTROLE-SEGMENTO-DO-USUaRIO

2.2. Localização de Pontos

Segundo FITZ, para determinar a localização de pontos na sua superfície, pode-se dividir a Terra em partes iguais, denominadas hemisfério.

De acordo com o sistema de convenção adotado, o Hemisfério Norte localiza-se ao norte da linha do Equador, e o Hemisfério Sul ao sul dessa mesma linha, e o Hemisfério Ocidental, a oeste do meridiano considerado como padrão, GREENWICH; e, finalmente, o Hemisfério Ocidental que passa a leste deste mesmo meridiano.

A figura a seguir ilustra os hemisférios:

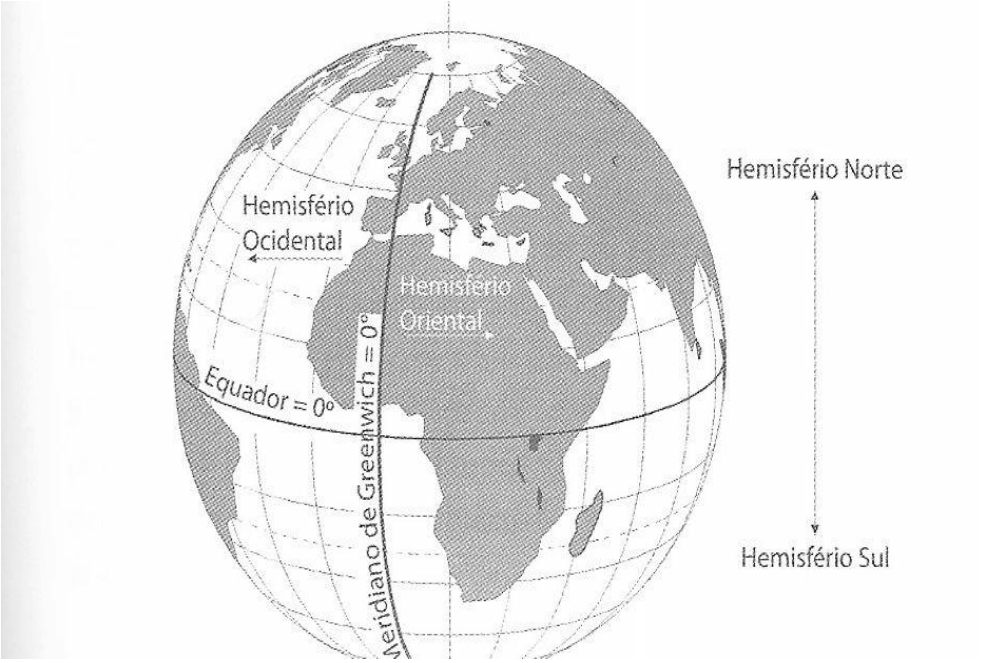

Figura 3: Hemisférios da Terra

Fonte: Fitz

2.2.1. SISTEMA DE COORDENADAS GEOGRÁFICAS

Ainda segundo FITZ, a forma mais usual de representar pontos num mapa é através da aplicação de um sistema sexagesimal, denominado de SISTEMA DE COORDENADAS GEOGRAFICAS.

Na figura que se segue pode-se observar como é representado as coordenadas geográficas:

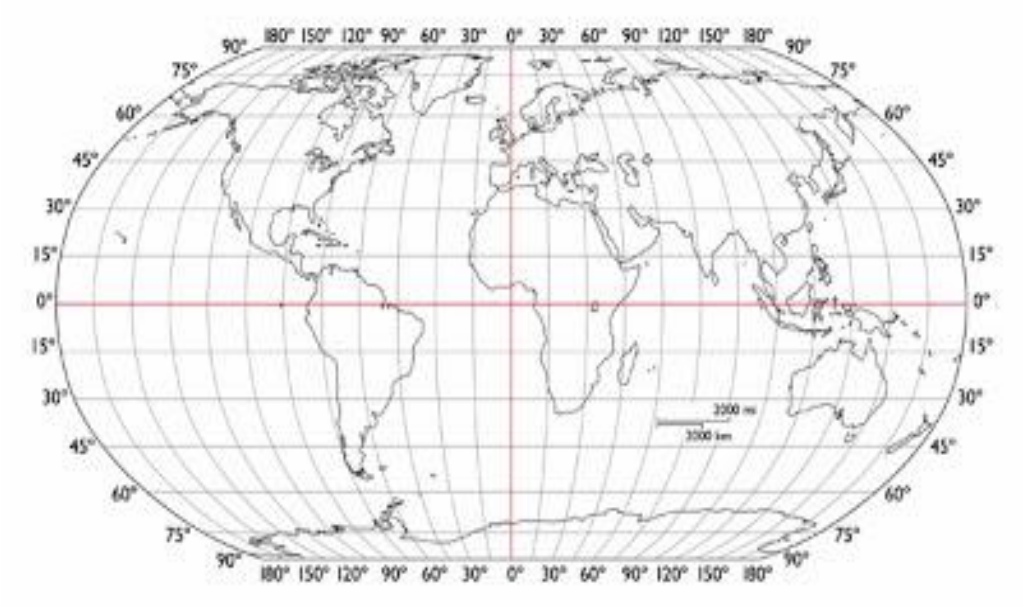

Figura 4: Coordenadas geográfica

Fonte: http://blog.cria.org.br/2013/10/coordenadas.html

Em que os valores dos pontos localizados na superfície terrestre são expressos por suas coordenadas geográficas, LATITUDE e LONGITUDE, contendo unidades de medida angular, ou seja, graus (º), minutos (') e segundos ('').

Em que latitude de um ponto é distancia angular entre o plano da superfície da terra, unido perpendicular ao centro do planeta, representado pela letra grega *fi* (φ), e com variação entre 0º e 90º, nas direções norte ou sul;

Longitude é o angulo formado entre o ponto considerado e o meridiano de origem (normalmente, Greenwich = 0º), e com variação entre 0º e 180º, nas direções leste ou oeste desse meridiano, representado pela letra grega *lambda* (λ).

Na figura que se segue é possível observar a representação dos conceitos acima.

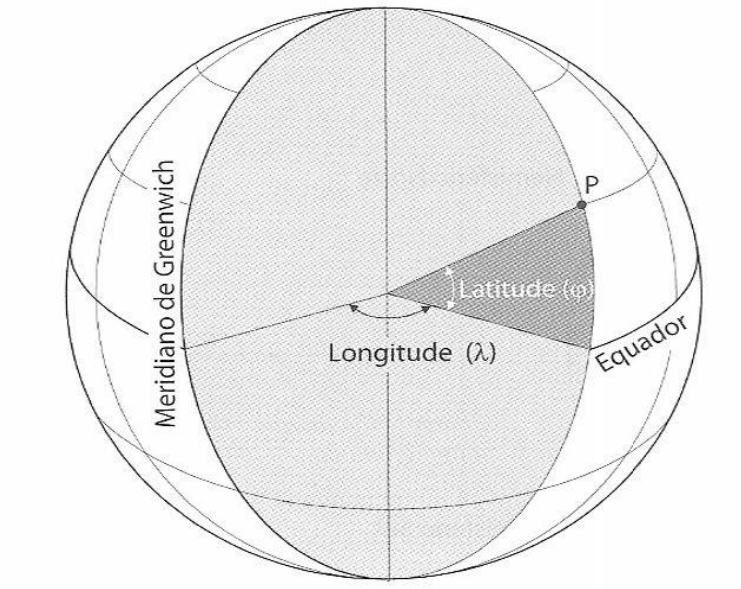

Figura 5: Latitude e Longitude

Fonte: Fitz

As COORDENADAS GEOGRAFICAS localizam, de forma direta, qualquer ponto sobre a superfícies terrestre, não havendo necessidade de qualquer outra indicação complementar.

Para isso, basta ser colocado, junto ao valor de cada coordenada, o hemisfério correspondente: N ou S, para a coordenada Norte ou Sul, e E ou W, para a coordenada Leste ou Oeste, respectivamente E de East (leste) e W de West (oest), podendo-se também utilizar L para Leste e O para Oeste. Pode-se utilizar, igualmente, os sinais + ou – para a indicação das coordenadas: N e E sinal positivo, e S e W sinal negativo.

Resumindo, quando o ponto estiver localizado ao sul do equador, a leitura da latitude será negativa, e a norte, positiva. Já com relação à longitude, quando o ponto estiver a oeste de Greenwich, seu valor será negativo, e a leste, positivo.

2.3. Computação móvel

De acordo com MATEUS e LOUREIRO (1998), a evolução tecnológica já chegou ao ponto onde é praticamente possível acessar informações em qualquer lugar do planeta em qualquer momento.

Segundo FIGUEIREDO e NAKAMURA (2003), tem-se notado também, uma grande evolução e popularização de dispositivos computacionais móveis, tais como celulares, PDAs (Personal Digital Assistants) e laptops, que traz-nos a estimativa de que em poucos anos milhares de pessoas espalhadas pelo mundo terão um desses tipos de dispositivos com a capacidade de comunicação com as redes fixas tradicionais e com outros computadores móveis. Esse ambiente propicia a criação do conceito de computação móvel.

Ainda de acordo FIGUEIREDO e NAKAMURA (2003), a computação móvel pode ser representada como um novo paradigma computacional que permite que utilizadores desse ambiente tenham acesso a serviços independentemente de sua localização, podendo inclusive, estar em movimento. Mais tecnicamente, é um conceito que envolve processamento, mobilidade e comunicação sem fio. A ideia é ter acesso à informação em qualquer lugar e a qualquer momento.

Para MATEUS e LOUREIRO (1998), a computação móvel surge como a quarta revolução da computação, antecedida pelos centros de processamento de dados da década de 60, o surgimento dos terminais nos anos setenta e as redes de computadores na década de 80. E o que define este novo paradigma é a mobilidade. Utilizadores podem acessar serviços independentemente de onde estejam localizados. Isso é possível graças a comunicação sem fio que elimina a necessidade de o utilizador manter-se conectado a infraestrutura física

2.3.1. DISPOSITIVOS PARA COMPUTAÇÃO MOVEL

Segundo FIGUEIREDO e NAKAMURA (2003), devido à definição de computação móvel, um dispositivo para este fim deve ter a capacidade de realizar processamento, trocar informações via rede e ser capaz de ser transportado facilmente pelo seu utilizador. Para isso, é importante que o dispositivo computacional tenha tamanho reduzido e não necessite de cabos para conectá-lo à rede de dados ou fonte de energia elétrica.

Assim, equipamentos deste tipo devem ter as seguintes características: ser bem menor que as estações de trabalho que costumamos usar, geralmente manipulados no colo ou na palma das mãos; possuir uma bateria, para evitar a necessidade de conexões à rede elétrica através de cabos que limitariam muito a mobilidade; e ter acesso a dados através de tecnologias de redes sem fio, pelo mesmo motivo anterior.

Exemplo de alguns dispositivos que têm sido usados para os fins da computação móvel, basicamente são os Smartphones, tablets, consolas de jogos portáteis, PDA's, computadores moveis, aparelhos de som portáteis, etc.

2.3.2. TIPOS DE APLICATIVOS MOVEIS

Segundo CHEDE (2013), basicamente temos três tipos de aplicativos (apps) moveis, cada um com suas potencialidades, mas também com restrições e cuidados especiais: os apps nativos, os apps web e os híbridos.

2.3.2.1. APLICATIVO NATIVOS

Um app nativo é focado em uma plataforma ou conjunto de dispositivos específicos e desenvolvido pelo modelo de programação desta plataforma.

Desenvolver um app nativo, ou seja, focado em uma plataforma especifica, significa que podemos explorar ao máximo a potencialidade da plataforma e de seus dispositivos. O app é um executável que é obtido através de download e executado no dispositivo e como resultado não pode ser executado em outro.

De maneira geral é obtido de um marketplace ou app store como exemplo a App Store da Apple ou Google Play para Android. Como o app tem acesso direto aos recursos da plataforma como câmeras, compasso, acelerômetro, calendário e lista de contatos etc., consegue-se tirar o máximo dos aparelhos. Também utiliza os componentes básicos de UI (User Interface) da plataforma, como botões e ícones, bem como suas interfaces gestuais.

2.3.2.2. APLICATIVO WEB

Os apps web são acessados via browsers, como muitas aplicações web do nosso velho conhecido desktop (ambiente de trabalho). Mas um app web móvel não é apenas pegar a aplicação que executa na web tradicional e deixa-la executar no smartphone ou tablet. O app deve reconhecer o ambiente em que ele está operando e ajustar-se de forma adequada. A tela que vai aparecer em um desktop deve ser diferente do que deve aparecer no tablet.

Entre os aspectos positivos temos a facilidade de desenvolvimento, uma vez que não demanda códigos únicos por plataforma, dado que são inerentemente cross-platform (reduz custos, naturalmente), e como é acessado via URL pode ser integrado a sistemas web como Twitter e Facebook, além de ter seu conteúdo acessado por motores de busca. Também bypassa as app stores, não precisando que o utilizador faça downloads.

2.3.2.3. APLICATIVO HIBRIDOS

Os apps híbridos basicamente contém dois elementos: uma componente web, baseado em HTML5, e um container ou “bridge” nativo, que permite acessar os recursos essenciais da plataforma e dispositivos.

O container constitui em um conjunto de APIs que permite o app HTML5 acessar os recursos nativos do smartphone ou tablet. O app hibrida tem seu código principal desenvolvido em HTML5 e é envelopado em um container, que o faz ser empacotado como um app nativo e, portanto, residindo em uma app store. Uma vantagem significativa é que as empresas podem usar o HTML5 para diminuir os custos de

desenvolvimento cross-platform, deixando apenas parte do código que explora a funcionalidade específica da plataforma ser escrita no modelo de programação da própria plataforma.

2.3.3. PRINCIPAIS PLATAFORMAS DE DESENVOLVIMENTO

Com o uso cada vez maior de dispositivos móveis, o número de plataformas e ambientes de desenvolvimento cresceu proporcionalmente.

A escolha de uma plataforma ideal para o desenvolvimento de um projeto significa optar por uma solução que forneça os melhores benefícios, em termos de custos, eficiência e tempo de desenvolvimento esperados para a finalização do projeto.

Segundo os dados das pesquisas realizadas pela INTERNATIONAL DATA CORPORATION (IDC), atualmente no mercado podemos encontrar várias plataformas para dispositivos moveis, mas as que mais se destacam são as plataformas Android, seguido do iOS e WindowsPhone.

2.3.3.1. ANDROID

O Android é o sistema operacional criado por Google, o seu código é aberto e, assim, acessível a todos os interessados. Ele foi baseado no núcleo Linux e suporta qualquer tipo de conexão sem fio - 3G, EDGE, Wi-Fi e Bluetooth). Ele é compatível com quase tudo, tratando-se de multimídia. Com a programação aberta ele é capaz de ser alterado, adaptar-se e pode manter um baixo custo. Ele já conta com inúmeros aplicativos para personalizar o seu Smartphone.
Esse tópico, no entanto, será abordado num tópico posterior visto que foi o Android a plataforma escolhida para a implementação do protótipo.

2.3.3.2. IOS

Segundo CAFE (2012), o sistema operacional iOS foi lançado em 2007 pela Apple e está presente na maioria dos dispositivos da empresa: iPod, iPhone e iPad. O iOS é baseado no Macintosh OS X, sistema operacional para plataforma desktop. Utiliza

como linguagem de programação o Objective-C, sua IDE é o XCode e é uma plataforma proprietária.

Segundo TAVARES (2013), o iOS só pode ser adquirido se você comprar um dispositivo da empresa que venha com o sistema. A Apple não permite que o iOS execute em aparelhos com hardware de terceiros, como acontece com os sistemas Android e o Windows Phone.

2.3.3.3. WINDOWS PHONE

Segundo JESUS (2014), o Windows Phone é uma versão móvel do Windows. Desenvolvido pela Microsoft, a plataforma foi lançada em 2010 como sucessora do Windows Mobile, de 2000, é baseado no Kernel do Windows CE6.

Para o desenvolvimento para Windows Phone é necessário ter alguma base de conhecimento na linguagem de programação C# ou VB, que são as duas possíveis linguagens de desenvolvimento. As ferramentas de desenvolvimento são o Visual Studio, e o SDK do Windows Phone. O SDK integra ao Visual Studio os templates de Windows Phone, as ferramentas no ToolBox e o Emulador usado para testar, simular e depurar suas aplicações, caso não tenha um dispositivo real.

2.3.4. TECNOLOGIA MOVEL NO MERCADO

Atualmente o utilizador tem as vantagens de acompanhar o crescimento tecnológico, o que os faz com que tornam-se cada vez mais exigentes quanto ás suas escolhas pois estes querem desfrutar máximos dessas vantagens.

É nesse contexto que os fabricantes veem-se na obrigação de conseguir melhorar as vantagens afim de seduzir e de conquistar o seu público-alvo e de garantir o seu futuro no mercado tudo isso têm-se desencadeado uma guerra tecnológica pela disputa de um ambiente competitivo que se abre para quem colocar no mercado, o produto mais atraente, interativo e versátil.

Segundo pesquisas da IDC, quase um terço da população mundial está conectada e chegamos a quase dois bilhões de usuários no mundo de dispositivos móveis que utilizam Android, iOS e Windows Phone, como pode-se observar no gráfico abaixo:

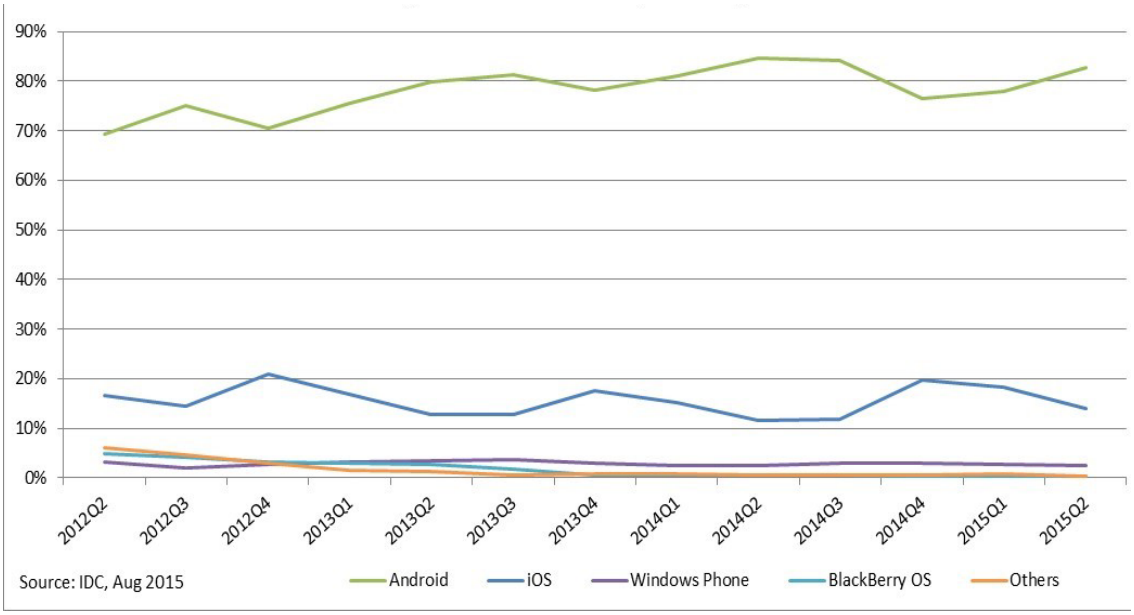

Figura 6: Gráfico do mercado Smartphone

Fonte: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

Período	Android	iOS	Windows Phone	Blackberry OS	Outros
2015Q2	82.8%	13.9%	2.6%	0.3%	0.4%
2014Q2	84.8%	11.6%	2.5%	0.5%	0.7%
2013Q2	79.8%	12.9%	3.4%	2.8%	1.2%
2012Q2	69.3%	16.6%	3.1%	4.9%	6.1%

Tabela 1: Tabela mercado Smartphone

Fonte: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

Em cabo verde de acordo com dados divulgados pela Agência Nacional das Comunicações (ANAC), no final do primeiro semestre, cerca de 634.889 de assinantes compunha o parque dos serviços móveis ativos associados a planos tarifários pós-pagos, pré-pagos, empresarias e clientes de banda larga móvel através de dongles/placas ou tablets sem voz.

De acordo com esses mesmos dados a taxa de penetração do serviço de telefonia móvel, no final do primeiro semestre de 2015, cresceu para 121 por 100 habitantes. Comparando esta taxa com a do período homólogo houve um crescimento de 15,9%. Excluindo os clientes que utilizam placas/dongles ou tablets sem voz, esta taxa seria de 112,8 por 100 habitantes.

De acordo com os dados estatísticos do o número de assinantes do serviço de telefonia móvel, corresponde a um aumento de 15% em relação ao primeiro semestre de 2014.

O número de assinantes do serviço de telefonia móvel em Cabo Verde segundo a ANAC, ultrapassa consideravelmente a população do país, o que indica que há utilizadores com mais de um cartão SIM (um circuito impresso utilizado para identificar, controlar e armazenar dados nos telefones celulares).

2.3.5. AREA DE TRANSPORTE

Diariamente podemos constatar que cada vez mais um elevado número de pessoas que deslocam-se para irem ao trabalho, escola, lazer, compras, etc., e muitos fazem isso utilizando os meios de transportes públicos.

Portanto, há que melhorar a segurança, eficiência e efetividade do transporte público, e aproveitando do rápido desenvolvimento tecnológico que se tem notado muitos países têm vindo a desenvolver e a implementar novam soluções tecnológicas mais precisamente para dispositivos moveis como forma de criar mais benefícios aos utilizadores dos transportes públicos bem como aos seus funcionários.

Benefícios esses que permitem minimizar o tempo de espera através de informações atualizadas e precisas sobre os horários, a segurança, a facilidades no pagamento de tarifas, informações sobre o transito, condições climáticas entre outros.

Em Cabo Verde atualmente os transportes públicos ainda não despõem de qualquer inovação tecnológicas que podem gerar esses tais benefício, a não ser os telemóveis pessoais dos condutores que permitem ao cliente que dispõem de seus números e efetuarem uma chamada perguntando ao condutor onde se encontram no momento e se podem ir busca-los a tal lugar.

2.4. Plataforma Android

Na computação móvel, a tecnologia Android está entre as mais avançadas em termos de mercado, funcionalidade e interface com o utilizador [Google 2012]. Esta seção irá descrever o surgimento da plataforma Android, bem como apresentar uma breve descrição técnica sobre a mesma.

2.4.1. HISTÓRIA E FILOSOFIA

O Android consiste em uma plataforma de desenvolvimento de aplicações para dispositivos móveis baseada em um sistema operacional Linux, com diversas aplicações já instaladas, além de oferecer também um ambiente de desenvolvimento poderoso e flexível [Lecheta 2010].

Ele foi lançado em 2007 pela Open Handset Allianse4 (OHA), a qual era um grupo recém-formado por fortes empresas do mercado de telefonia e liderado pelo Google. Na época a OHA era constituída por cerca de 30 integrantes, mas hoje já conta com cerca de 90 empresas. O objetivo deste grupo era criar uma plataforma única e aberta para celulares e que pudesse adequar aos interesses dos utilizadores comuns, dos desenvolvedores e dos ambientes corporativos.

Devido ao fato de que esta plataforma é baseada em princípios open source, ela vem apresentando excelentes resultados e ótimas perspectivas de aperfeiçoamento. A sua

filosofia aberta permite que as empresas fabricantes de Smartphones customizem o Android sob a perspectiva que convir-lhes, bem como facilita para que desenvolvedores do mundo todo contribuam corrigindo bugs, adicionando novas funcionalidades ou desenvolvendo novas aplicações [Lecheta 2010].

2.4.2. DEFINIÇÃO

Segundo GADELHA (2011), Android é um sistema operacional para dispositivos móveis, baseado em uma plataforma de código aberta sob a licença apache, permitindo que os fabricantes possam modificar seu código fonte, essa liberdade permite adicionar novos recursos e incorporá-los ao sistema, desta forma o software evolui constantemente.

O sistema possui cerca de 12 milhões de linhas de código, sendo: 3 milhões em XML; 2.8 milhões em C; 2.1 milhões em Java; e 1.75 milhões em C++.

Android oferece uma abordagem unificada para o desenvolvimento de aplicativos para dispositivos móveis que significa que os desenvolvedores só precisam desenvolver para o Android, e suas aplicações devem ser capazes de rodar em diferentes dispositivos alimentados por Android.

2.4.3. ARQUITETURA DO ANDROID

A descrição deste tópico baseia-se no artigo publicado por TOSIN (2011) em que acordo com a figura pode-se verificar que a arquitetura do Android é composta por várias camadas e componentes.

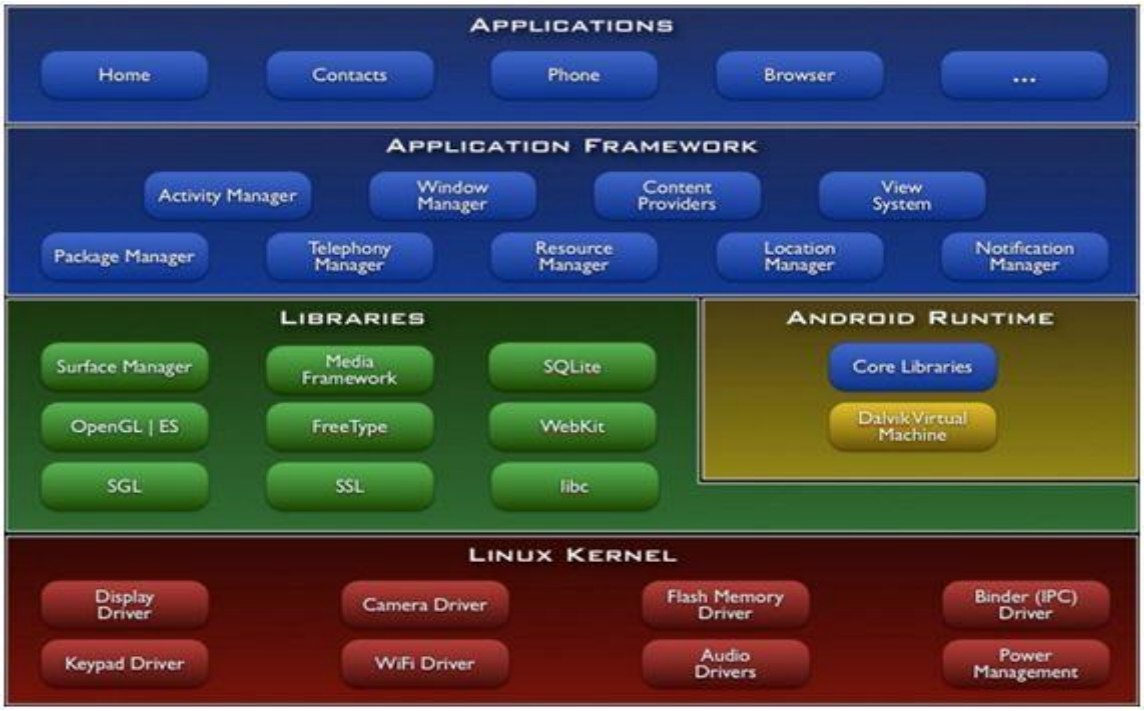

Figura 7: Arquitetura plataforma Android

Fonte: http://www.softblue.com.br/blog/home/postid/11/CONHECENDO+O+ANDROID

2.4.3.1. APLICATIONS

Esta é a camada das aplicações em que podemos encontrar as funções básicas do dispositivo, que permite a interação entre o dispositivo e o utilizador, nela podemos encontrar os gerenciadores de contatos, os navegadores, calculadora, agenda, etc.

2.4.3.2. APLICATIONS FRAMEWORK

Na camada de Framework encontramos as API's que são as responsáveis para o gerenciamento das aplicações básicas da plataforma. O desenvolvedor tem o acesso total ao framework e os conjuntos de ferramentas básicas que as constituem permitindo desenvolver outras ferramentas mais complexas de acordo com as suas necessidades. Podemos encontrar nesta camada os serviços de localização, de notificação, provedores de conteúdo, etc.

2.4.3.3. LIBRARIES

A camada Livraria é a camada onde podemos encontrar um conjunto de bibliotecas C/C++ que são usados pelos diversos componentes do sistema, como o OpenGL ES (para renderização 3D), SQLite (gerenciador de bancos de dados) e suporte a diversos formatos de áudio e vídeo, etc.

2.4.3.4. ANDROID RUNTIME

Nesta camada encontramos um conjunto de bibliotecas do núcleo Java (Core Libraries), e também podemos encontrar a Máquina Virtual Dalvik (DVM) que permite executar cada aplicação com seu próprio processo, o que irá permitir que múltiplas instâncias executem ao mesmo tempo.

2.4.3.5. LINUX KERNEL

Nesta camada encontra-se localizada o sistema operacional da plataforma que é baseada no Linux, nela podemos encontrar os programas de gerenciamento de memória, gerenciamento de processos, configurações de segurança e vários drivers de hardware.

2.4.4. COMPONENTES DE UM APLICATIVO ANDROID

Para que o desenvolvedor chega ao resultado esperado ao desenvolver uma aplicação é necessário compor uma serie de componentes.

A figura 8 mostra esses componentes.

Figura 8: Componente de um aplicativo Android

Fonte: http://www.softblue.com.br/blog/home/postid/11/CONHECENDO+O+ANDROID

2.4.4.1. ACTIVITIES

É a representação das telas das aplicações do dispositivo, permitindo a interação com o utilizador. Possui interface composta por Views, componentes gráficos, eventos e etc.

2.4.4.2. SERVICES

São serviços ou códigos sem interfaces para o utilizador, executam em segundo plano, possuem ciclo de vida próprio.

São serviços que não necessitam de interação direta com o utilizador e normalmente utilizam um grande tempo de execução. Exemplo: Envio da atualização da posição do utilizador para um servidor remoto.

2.4.4.3. CONTENT PROVIDERS (PROVEDORES DE CONTEÚDO)

É a forma utilizada pela plataforma para compartilhar dados entre os aplicativos. Exemplo: o gerenciamento de contato do Android é uma aplicação nativa do sistema, mas permite que aplicações desenvolvidas por terceiros acessem e atualizem os contatos.

2.4.4.4. BROADCAST RECEIVERS

São componentes com responsabilidades de receber e tratar eventos (ou broadcast). Os eventos podem ser gerados por aplicações nativos do sistema ou por outras aplicações. Exemplo: a indicação que a carga de bateria está fraca, a indicação que o carregador acabou de ser ligado, indicação do recebimento de uma chamada ou de um SMS.

2.4.4.5. ANDROIDMANIFEST.XML

É um arquivo XML único e obrigatório para cada aplicativo, nele são feitas as configurações gerais da aplicação e dos seus componentes, como configurações sobre permissões e Services necessárias para o funcionamento do sistema, informações sobre as Activities.

2.4.5. CICLO DE VIDA DE UM APLICATIVO ANDROID

Segundo GOOGLE (2012) um Activity é uma componente base de um aplicativo Android, a ele está sempre associado, a ela está sempre associado um layout na qual o utilizador interage com a aplicação. E cada Activity possui um ciclo de vida que é gerenciado pelo sistema operacional.

A figura 9 ilustra o ciclo de vida de um aplicativo Android.

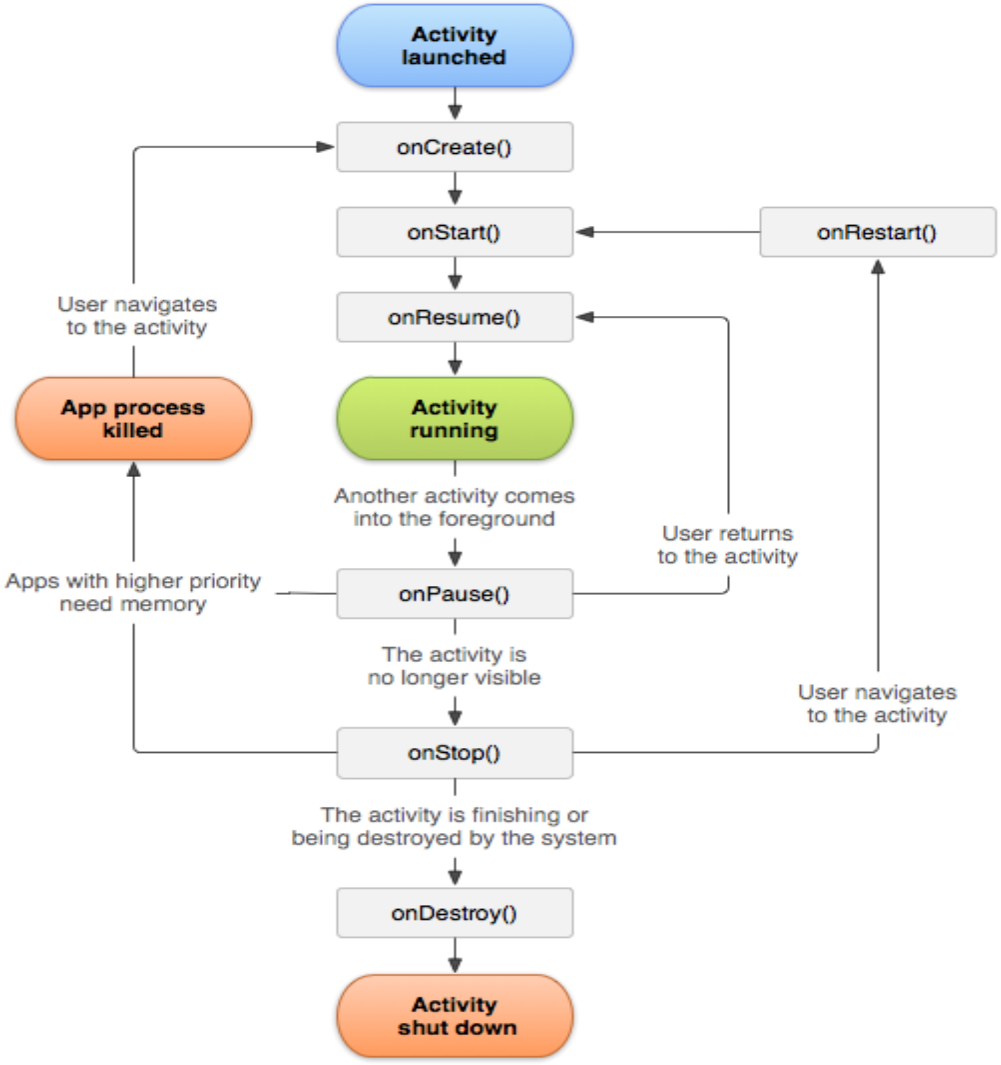

Figura 9: Ciclo de vida de um aplicativo Android

Fonte: http://developer.android.com/guide/components/activities.html

Para cada fase do ciclo de vida de uma Activity, o Android oferece um método que pode ser sobrescrito e realizar os tratamentos necessários. A seguir serão listados e descritos os métodos dos ciclos de vida de uma Activity segundo DA SILVA (2014).

onCreate() - É a primeira função a ser executada em uma Activity. Geralmente é a responsável por carregar os layouts XML e outras operações de inicialização. É executada apenas uma vez.

onStart() - É chamada imediatamente após a onCreate() – e também quando uma Activity que estava em background volta a ter foco.

onResume() - Assim como a onStart(), é chamada na inicialização da Activity e também quando uma Activity volta a ter foco. Qual a diferença entre as duas? A onStart() só é chamada quando a Activity não estava mais visível e volta a ter o foco, a onResume() é chamada nas "retomadas de foco".

onPause() - É a primeira função a ser invocada quando a Activity perde o foco (isso ocorre quando uma nova Activity é iniciada).

onStop() - Só é chamada quando a Activity fica completamente encoberta por outra Activity.

onRestart() - Chamada imediatamente antes da onStart(), quando uma Activity volta a ter o foco depois de estar em background.

onDestroy() - A última função a ser executada. Depois dela, a Activity é considerada "morta" – ou seja, não pode mais ser relançada. Se o utilizador voltar a requisitar essa Activity, um novo objeto será construído.

2.4.6. MAPAS E GPS

Uma das funcionalidades que mais chamam a atenção na plataforma Android é a relativamente fácil integração de uma aplicação com o Google Maps [Lecheta 2010]. Para tanto, o Google oferece um pacote de serviços para o Android, no qual está incluída a Google Maps API (Application Programming Interface).

Atualmente a API do Google Maps encontra-se na versão 2, e fornece uma série de recursos nativos e atualizados para o desenvolvedor (como por exemplo zoom e navegação pelo mapa). Ela também permite ao desenvolvedor implementar algumas customizações no mapa, como por exemplo adicionar objetos e marcações ao mapa, bem como oferece a possibilidade de interações com o utilizador a partir dessas marcações [Google 2013].

3. ANALISE DO SISTEMA A DESENVOLVER

Neste capítulo irei abordar sobre a arquitetura proposto para o sistema, a modelagem do sistema feita através de anotações gráficas de diagramas UML e a análise dos requisitos do sistema.

3.1. Arquitetura da solução proposta

De acordo com os objetivos estipulados para o referido projeto o sistema será composto por um servidor Web que aloja a parte do Web Service desenvolvido em PHP e da base de dados em MySQL onde posteriormente serão tratados os dados do sistema.

A aplicação cliente para ter acesso à sua localização utilizará o sistema de GPS do dispositivo, o mapa será disponibilizado pelos serviços da Google onde será necessário ter uma conexão com a internet tanto para obter os serviços do mapa bem como também para ter acesso ao envio e recepção dados alojados no servido Web.

A figura a seguir demostra a arquitetura proposto para o sistema.

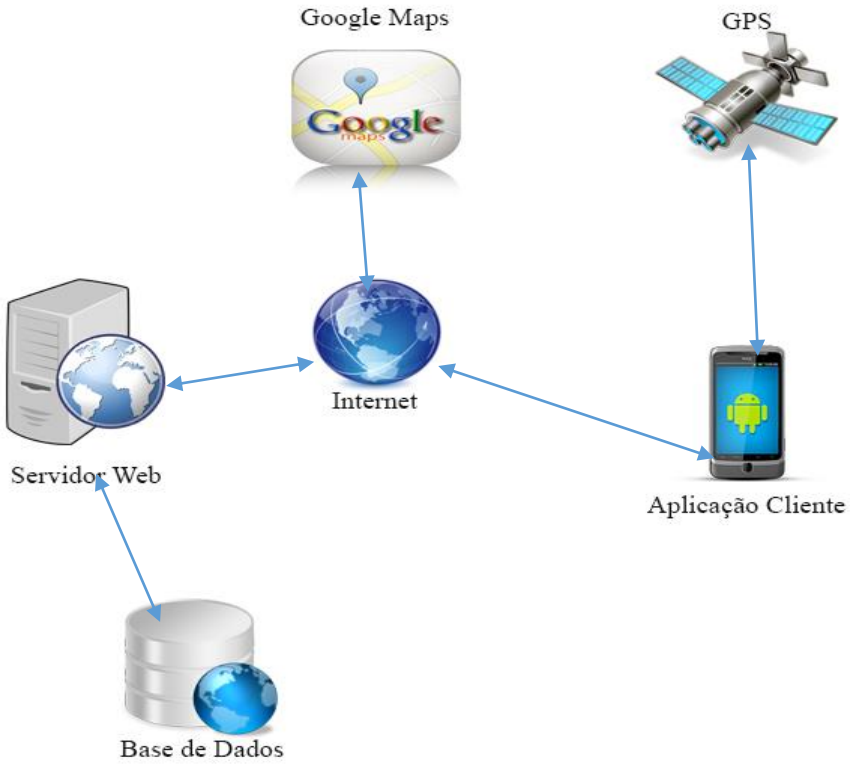

Figura 10: Arquitetura do sistema

Fonte: Elaborado pelo autor

3.2. Modelagem do sistema

Para fazer a modelação do Sistema foi utilizado uma notação gráfica standard – UML (Unified Modelling Language), que pode ser traduzida por Linguagem de Modelação Unificada.

Segundo NUNES e O'NEILL (2004), a UML é uma linguagem que utiliza uma notação padrão para especificar, construir, visualizar e documentar sistemas de informação orientado por objetos.

A UML funciona como meio de comunicação entre os diversos elementos envolvidos no processo, utilizadores, gestores e equipa de desenvolvimento. Pode ser utilizada para documentar o sistema ao longo de todo o ciclo de desenvolvimento, começando

com a tarefa inicial de análise dos processos de negócio da organização e prolongando-se até à tarefa de manutenção evolutiva do sistema informático.

Ainda segundo os mesmos um modelo em UML é constituído por um conjunto de diagramas que representam aspectos complementares de um sistema de informação, dentre as quais serão abordados:

- Diagrama de Use Cases;
- Diagrama de Classes;
- Diagrama de Sequência;
- Diagrama de Atividade;
- Diagrama de Componente;

3.2.1. IDENTIFICAÇÃO DOS ATORES

Segundo NUNES e O'NEILL (2004), um ator representa uma entidade externa que interage com o sistema.

Para o referido projeto teremos os seguintes atores:

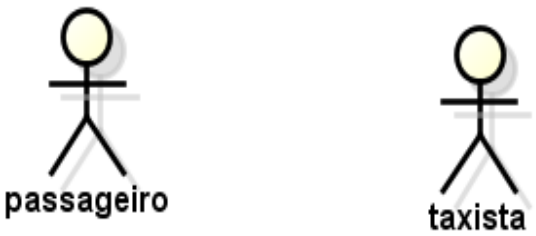

Figura 11: Atores do sistema

Fonte: Elaborado pelo Autor (2015)

De acordo com NUNES e O'NEILL (2004), apesar da representação humanizada, os atores podem não ser só pessoas, mas também outros sistemas físicos ou lógicos como, por exemplo, um modulo de Contabilidade.

Os atores devem ser caracterizados através de uma pequena descrição, de forma a assegurar uma correta compreensão do significado do ator por todos os elementos da equipa envolvida na análise.

No quadro a seguir pode-se constatar a descrição do ator passageiro

Ator	Passageiro
Definição	O Passageiro é a parte interessada em obter informações táxis ao seu redor. Ele é capaz de visualizar informações gerais sobre cada táxi de seu interesse, assim como informações sobre a distância e de requisitar os seus serviços através de uma chamada telefônica o de um SMS.
Frequência de uso	Diário
Conhecimento em informática	O necessário para manusear um Smartphone
Grau de escolaridade	Ensino Básico a ensino superior
Permissão de acesso	Acesso a informações como: posição geográfica dos táxis, dados de cada taxista, possibilidade de realizar chamadas telefônicas e envio de SMS.

Tabela 2: Descrição ator passageiro

Fonte: Elaborado pelo autor

No quadro a seguir pode-se constatar a Descrição do ator Taxista

Ator	Taxista
Definição	O Taxista é a parte interessada em fornecer informações sobre a sua localização geográfica. Ele é capaz de registrar e fazer entrar no sistema fornecer dados, ativar o envio automático de informações sobre a sua localização geográfica, de alterar o seu status e atender pedidos através de uma chamada telefônica o de um SMS.
Frequência de uso	Diário
Conhecimento em informática	O necessário para manusear um Smartphone
Grau de escolaridade	Ensino Básico a ensino superior
Permissão de acesso	Permite o aceso à sua posição geográfica, acesso a seus dados, permissão de atender pedidos por chamadas telefônicas e SMS.

Tabela 3: Descrição ator Taxista

Fonte: Elaborado pelo autor

Depois de identificar os atores do sistema, já é possível determinar os requisitos essenciais para a elaboração do sistema bem como os uses cases.

3.2.2. ANÁLISES DE REQUISITOS DO SISTEMA

Os requisitos do sistema subdividem em funcionais e não funcionais:

Requisitos funcionais - são requisitos do sistema que especificam funções que o sistema ou componente deve ser capaz de realizar, definem o comportamento do sistema, ou seja o processo ou transformação que componentes de software ou

hardware efetuam sobre as entradas para gerar as saídas. Esses requisitos capturam as funcionalidades sob o ponto de vista do utilizador.

Requisitos não funcionais - são os requisitos que não estão diretamente relacionados à funcionalidade do sistema, ou seja, são as propriedades e características desejadas do sistema relativas à capacidade de armazenamento, tempo de resposta, configuração, uso (ex. uso intuitivo), confiabilidade, etc.

3.2.2.1. REQUISITOS FUNCIONAIS DO APLICATIVO PASSAGEIRO

Mostra lista dos táxis disponíveis – o sistema deverá disponibilizar uma lista dos taxis que se encontram disponíveis mais próximos do utilizador bem como a suas respectivas informações.

Apresentar informações e o Mapa – o sistema deverá apresentar um mapa permitindo ao utilizador visualizar a sua posição atual bem como a dos veículos que se encontram próximos a ele e a respectiva rota e as suas informações.

Requisitar táxis – o sistema deverá permitir ao utilizador a requisição do taxi pretendido através de uma chamada telefónica ou de envio de um SMS.

3.2.2.2. REQUISITOS FUNCIONAIS DO APLICATIVO TAXISTA

Registrar usuário - o sistema deverá permitir o registo de utilizadores, informando o nome, o telefone, o e-mail, o nome de utilizador e a senha.

Realizar Login - depois do utilizador estiver cadastrado o sistema deverá permitir o login do utilizador através do seu nome de utilizador e da sua senha.

Editar informações do veículo - o sistema deverá permitir que o utilizador introduza as informações do veículo que ele estiver a trabalhar no momento.

Alterar o estado do veículo - o sistema deverá permitir ao utilizador alterar o seu estado podendo este estar ocupado ou livre.

Atender pedidos **–** o sistema deverá permitir ao utilizador atender pedidos através de chamadas telefônicas e SMS.

Enviar coordenadas geográficas ao servidor **–** o sistema deverá permitir que o utilizador ative e desative a opção de enviar coordenadas geográfica automaticamente a um servidor remoto num período de cinco em cinco segundos e em um metro percorrido.

3.2.2.3. REQUISITOS NÃO FUNCIONAIS

Neste tópico serão mostrados os requisitos não-funcionais do sistema desta pesquisa, agrupados pela sua classificação.

Usabilidade **-** o sistema deverá apresentar os textos na língua Portuguesa, Inglesa e Francesa. Deverá permitir que a tela do dispositivo seja usada tanto na horizontal como na vertical, a resolução mínima permitida pelo sistema é de 240 X 320, durante o processamento dos dados deverá mostrar uma janela com a mensagem "Carregando", caso o utilizador não dispor de uma conexão com a internet o sistema deverá notificar o mesmo.

Operacionais **-** o sistema deverá operar na plataforma Android 2.3 ou superior, obrigatório o uso de internet, deverá usar o GPS do dispositivo para fornecer informações sobre a posição do utilizador e utilizará os serviços do Google Maps para visualizar as localizações no mapa.

Implementação **-** o modulo cliente será desenvolvido utilizando a linguagem de programação Java, o modulo servidor utilizará a linguagem PHP e o banco de dados utilizará a linguagem SQL.

Portabilidade – o sistema poderá ser transportado e instalado em qualquer aplicativo capaz de operar utilizado plataforma Android;

Facilidade e confiabilidade – O sistema deverá ser de fácil utilização para os utilizadores, e deverá transmitir credibilidade aos mesmos.

Arquitetura **-** a camada servidor deverá operar utilizando o sistema operacional Windows, a camada cliente utilizará o sistema operacional Android, o servidor deverá possuir no mínimo 512 *megabytes* de memória RAM, prevendo o crescimento do sistema a arquitetura do sistema deverá ser escalável ou seja deverá possuir a capacidade de aumentar o seu poder de processamento e armazenamento de dados sempre que necessário face ao crescimento dos utilizadores, o SGBD utilizado para gerenciar os dados do sistema será o MySQL, o webservice e o banco de dados serão suportados pelo servidor Apache, a comunicação entre a camada utilizador e a camada servidor utilizará o padrão de comunicação REST, utilizando o protocolo HTTP, as mensagens trocadas entre as camadas seguirão o padrão do formato JSON;

3.2.3. DIAGRAMAS DE CASO DE USO

Segundo SILVA e VIDEIRA (2001), um diagrama de casos de utilização descreve a relação entre atores e casos de utilização de um dado sistema permitindo dar uma visão global e de alto nível do sistema.

Na Figura 12, é apresentado o diagrama de caso de uso resultante do processo de planejamento do projeto, na visão do ator para o usuário Passageiro.

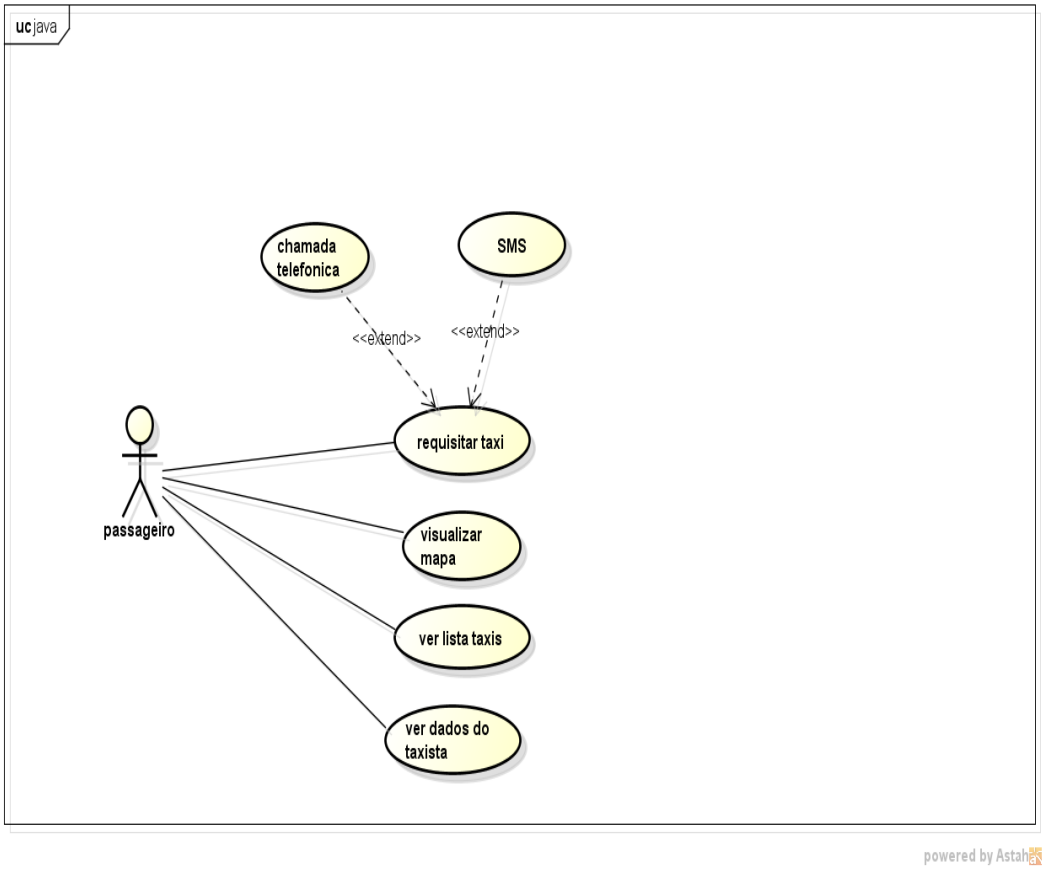

Figura 12: Diagrama Use Case Passageiro

Fonte: Elaborado pelo autor

Na Figura 13, encontram-se os casos de uso para o usuário Taxista.

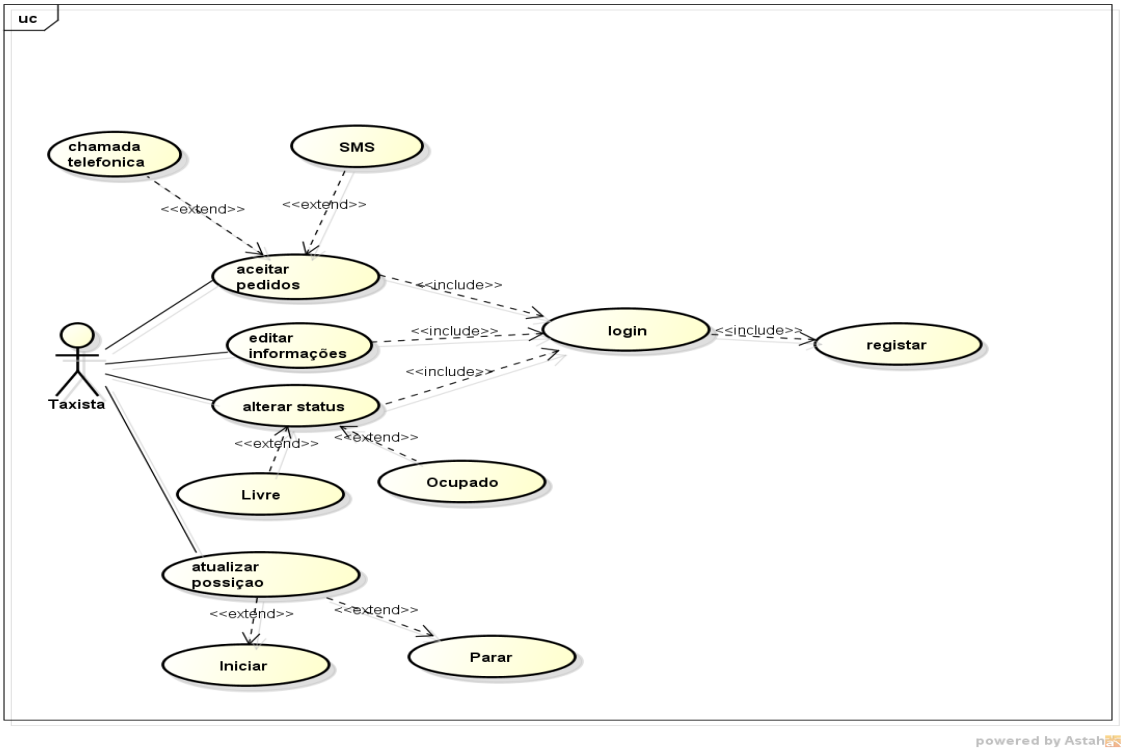

Figura 13: Diagrama Use Case Taxista

Fonte: Elaborado pelo autor

3.2.4. DIAGRAMA DE CLASSES

De acordo com SILVA e VIDEIRA (2001), uma classe é a descrição de um conjunto de objetos que partilham os mesmos atributos, operações, relações e a mesma semântica. Uma classe corresponde a algo tangível ou a uma abstração conceptual existente no domínio do utilizador ou no domínio do engenheiro de software.

Ainda segundo os mesmos, uma classe bem estruturada é simples e facilmente entendida; providencia uma abstração definida a partir do vocabulário do domínio do problema ou do domínio da solução; agrega um conjunto restrito e bem definido de responsabilidades; e providencia uma separação clara entre a especificação abstrata e a sua implementação.

As figuras seguintes demostram respetivamente o diagrama de classes do sistema Taxista e do Passageiro:

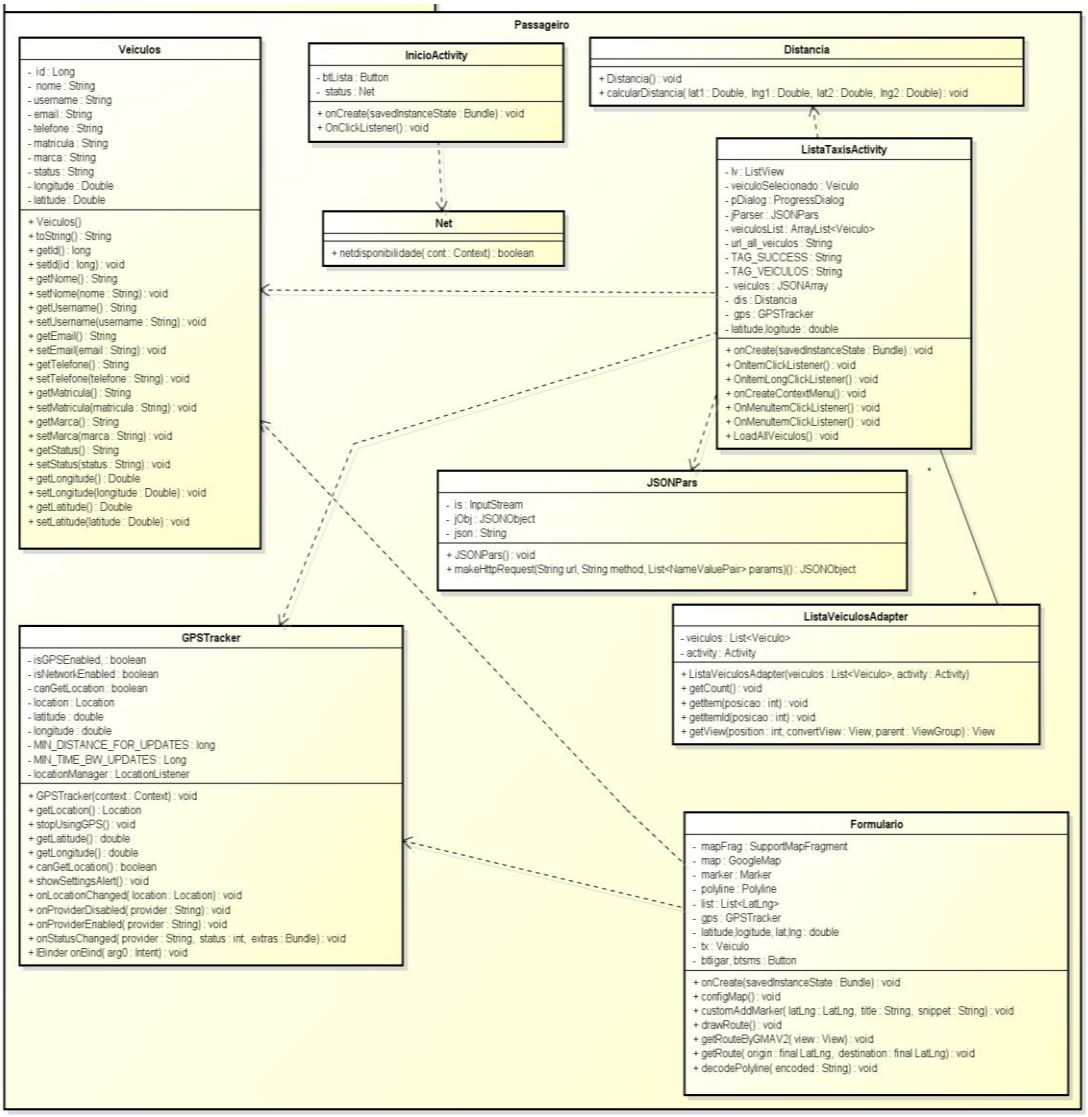

Figura 14: Diagrama de Classe Passageiro

Fonte: Elaborado pelo autor

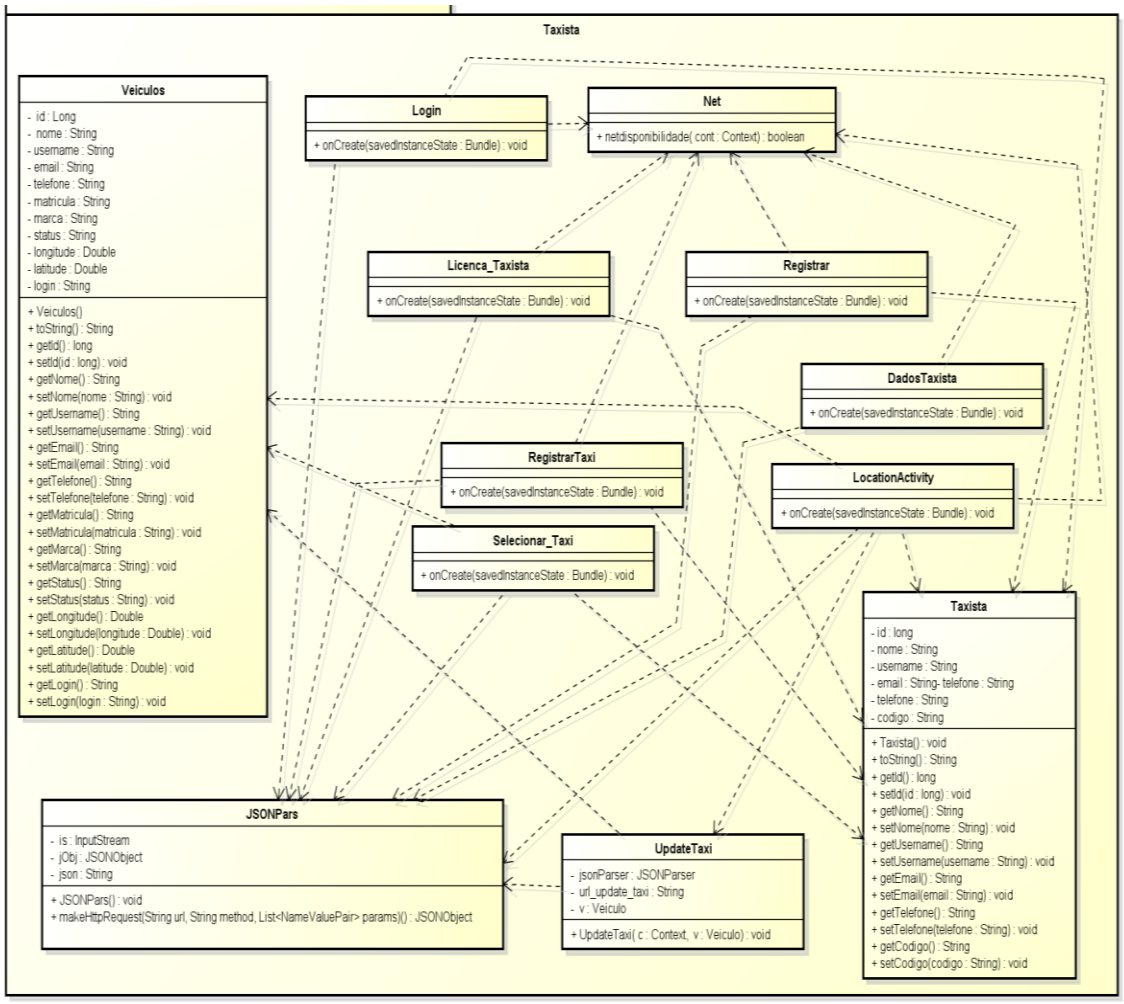

Figura 15: Diagrama de Classe Taxista

Fonte: Elaborado pelo autor

3.2.5. DIAGRAMA DE ATIVIDADE

De acordo com NUNES e O'NEILL (2004), o diagrama de atividades constitui um elemento de modelação simples, mas eficaz para descrever fluxos de trabalho numa organização ou para detalhar operações de uma classe, incluindo comportamentos que possuam processamento paralelo.

Ainda segundo os mesmos um diagrama de atividades é a capacidade de descrever conjuntos de atividades que se desenvolvem em paralelo. Esta capacidade pode ser utilizada, por exemplo, quando se descreve um projeto de desenvolvimento de

software, no qual algumas das atividades podem ser realizadas em simultâneo por diversos atores.

As figuras seguintes demostram respetivamente o diagrama de atividade do sistema para o Taxista e para o Passageiro:

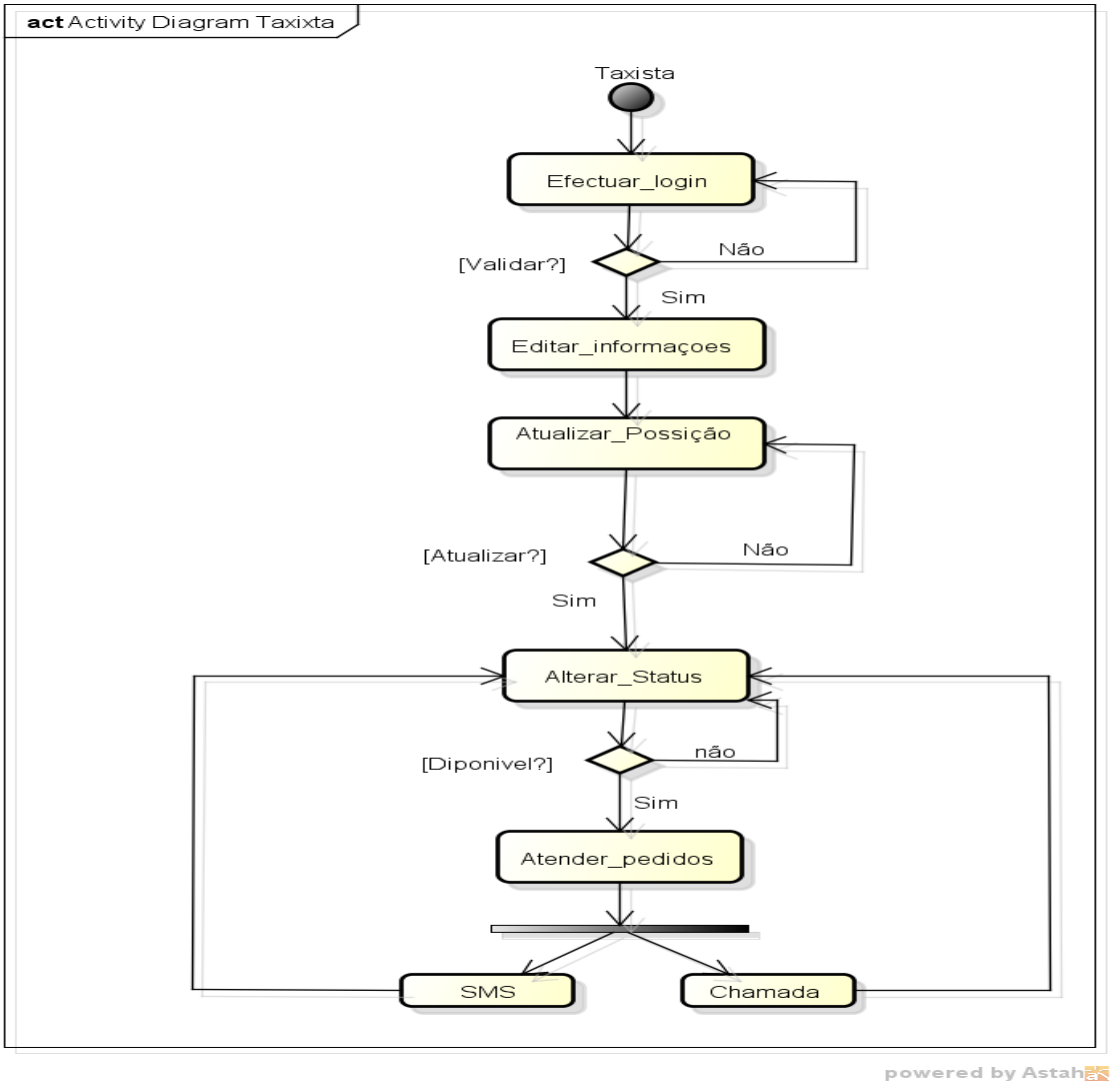

Figura 16: Diagrama de Atividade Taxista

Fonte: Elaborado pelo autor

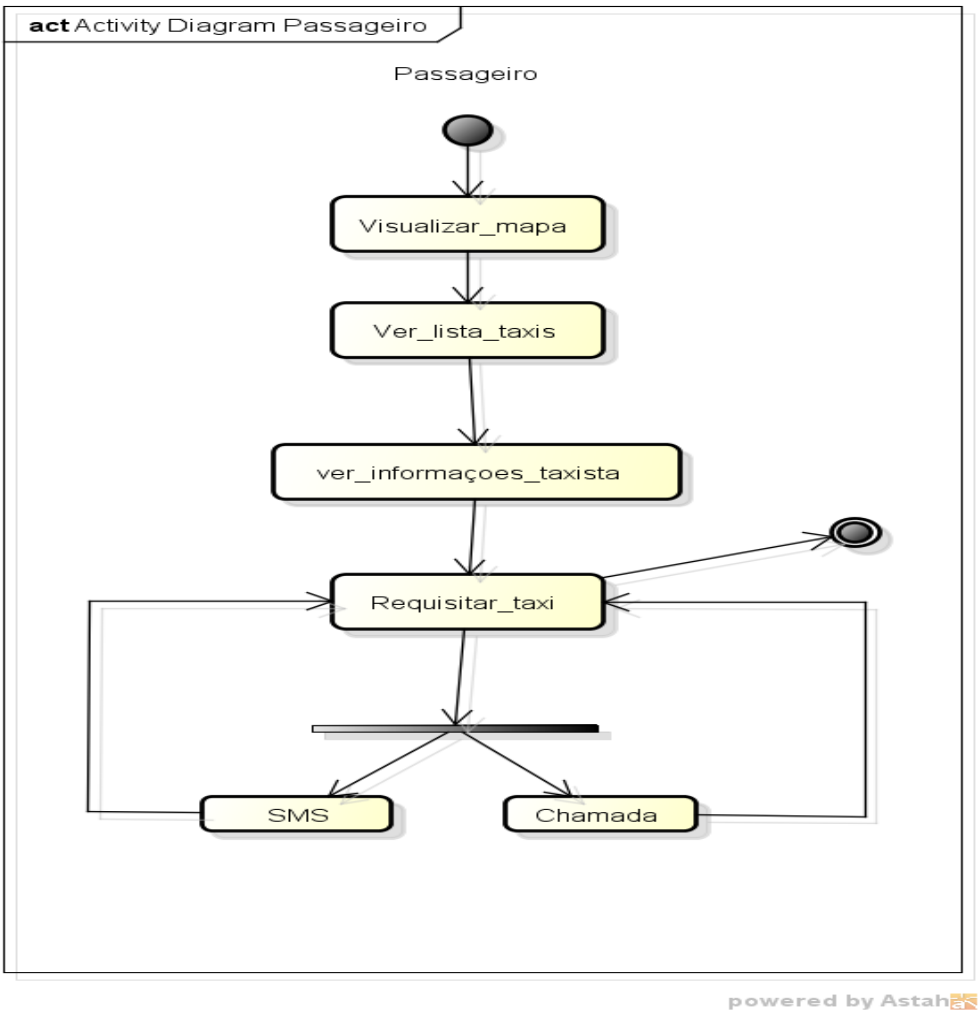

Figura 17: Diagrama de Atividade Taxista

Fonte: Elaborado pelo autor

3.2.6. DIAGRAMA DE SEQUENCIA

Segundo SILVA e VIDEIRA (2001), o diagrama de sequência ilustra uma interação segundo uma visão temporal. Um diagrama de sequência é representado através de duas dimensões: a dimensão horizontal, que representa o conjunto de objetos intervenientes; e a dimensão vertical que representa o tempo.

Para NUNES e O'NEILL (2004), o diagrama de sequência é um diagrama de interação que realça a ordem cronológica das mensagens entre objetos.

As figuras seguintes demostram respetivamente o diagrama de sequência do sistema para requisitar um taxi por parte do Passageiro e para atender uma solicitação por parte do Taxista:

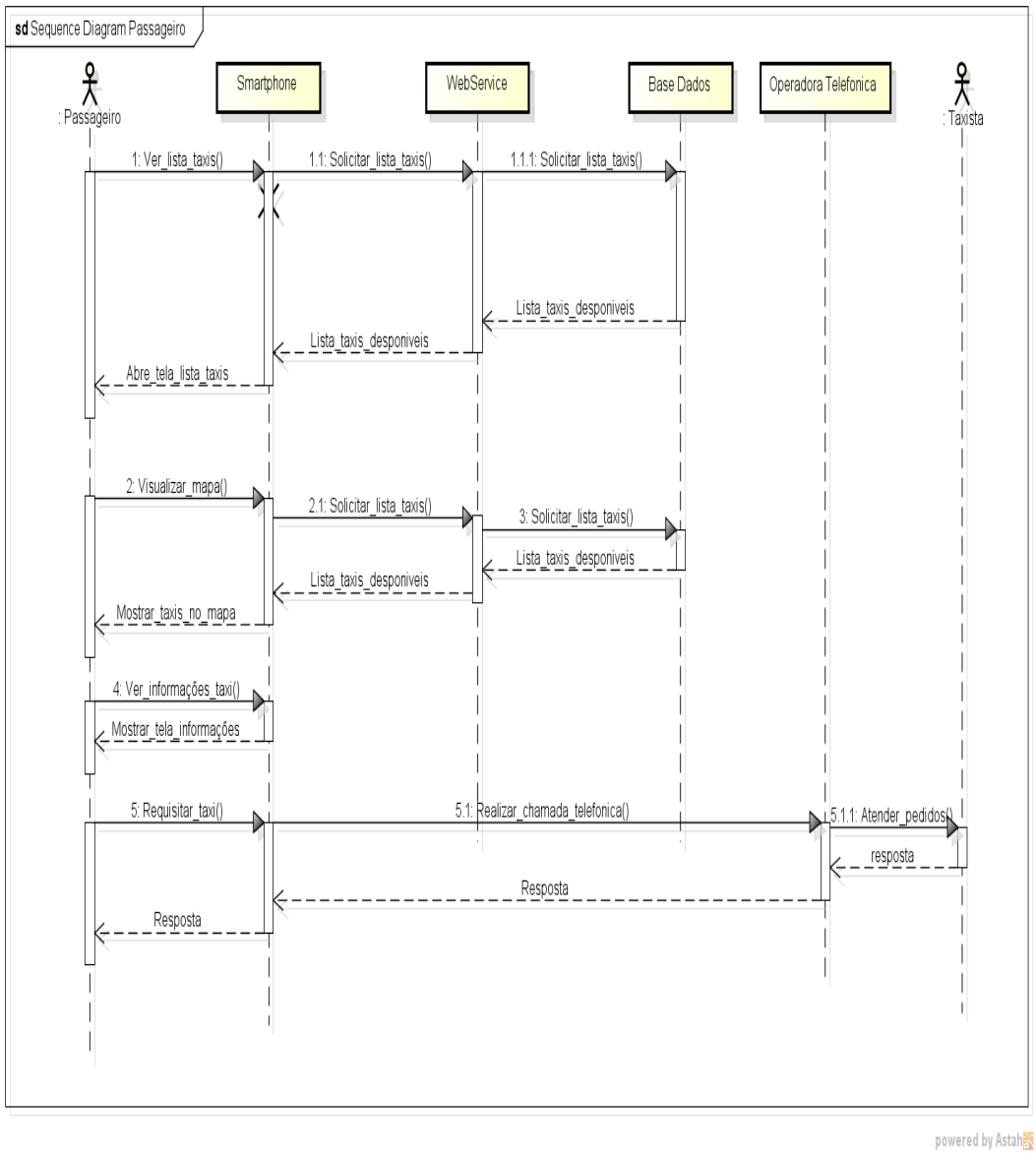

Figura 18: Diagrama de Sequencia requisitar taxi

Fonte: Elaborado pelo autor

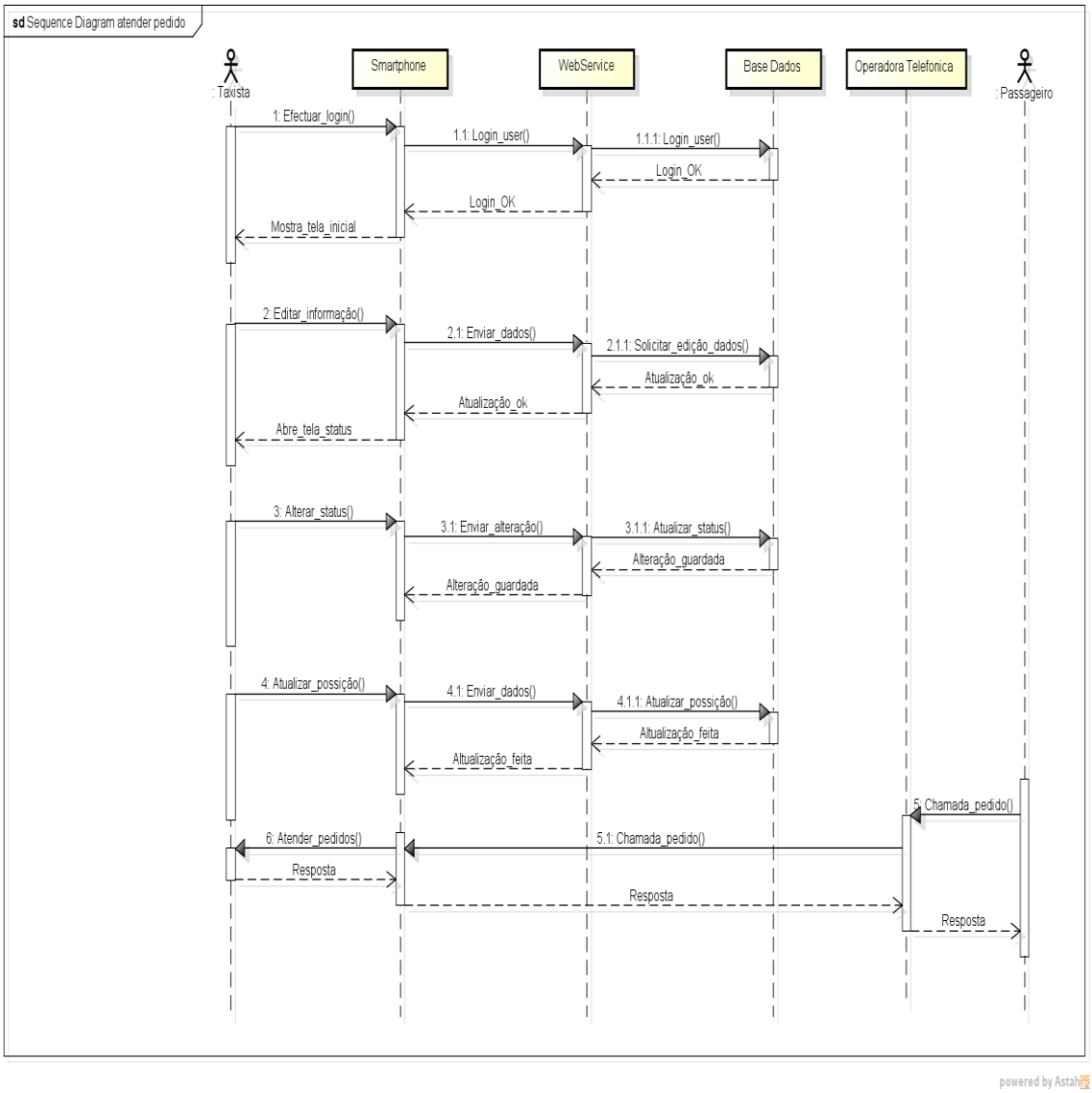

Figura 19: Diagrama Sequencia atender solicitação

Fonte: Elaborado pelo autor

3.2.7. DIAGRAMA COMPONENTE

Segundo NUNES e O'NEILL (2004), um diagrama de componentes mostra um conjunto de componentes e as suas relações.

Um componente representa um módulo físico de código, sendo o resultado do desenvolvimento numa linguagem de programação ou outra técnica. O

desenvolvimento por componentes permite reforçar a reutilização como forma de diminuição de custos e possíveis erros, dado que podem ser previamente já testados.

Na figura a seguir é demostrada o diagrama de componente proposto para o desenvolvimento do sistema:

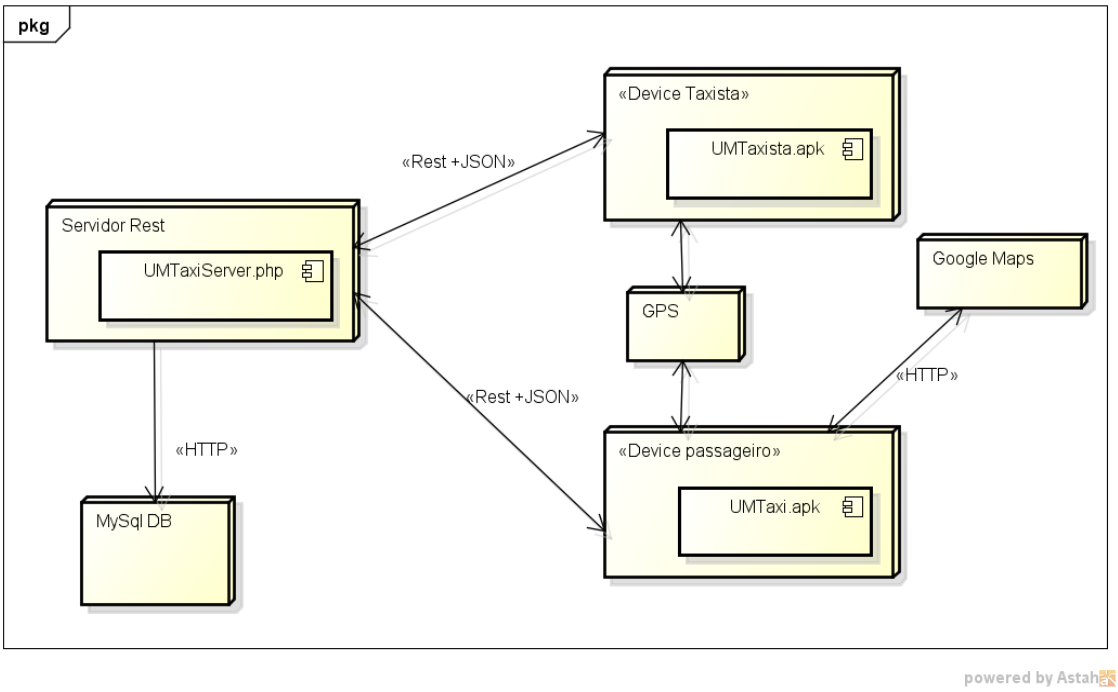

Figura 20: Diagrama de componente

Fonte: Elaborado pelo autor

Normalmente, os componentes encontram-se interligados por uma relação de dependência, que demonstra o impacto nos diversos componentes das alterações a um componente em particular.

3.2.8. BASE DE DADOS

Para o armazenamento dos dados referentes à aplicação, foi utilizado o SGBD MySql para a criação de uma base de dados. A base de dados encontra-se armazenada num servidor online do provedor de hospedagem Hostinger. As tabelas da base de dados podem ser vistas no diagrama apresentado na figura 21.

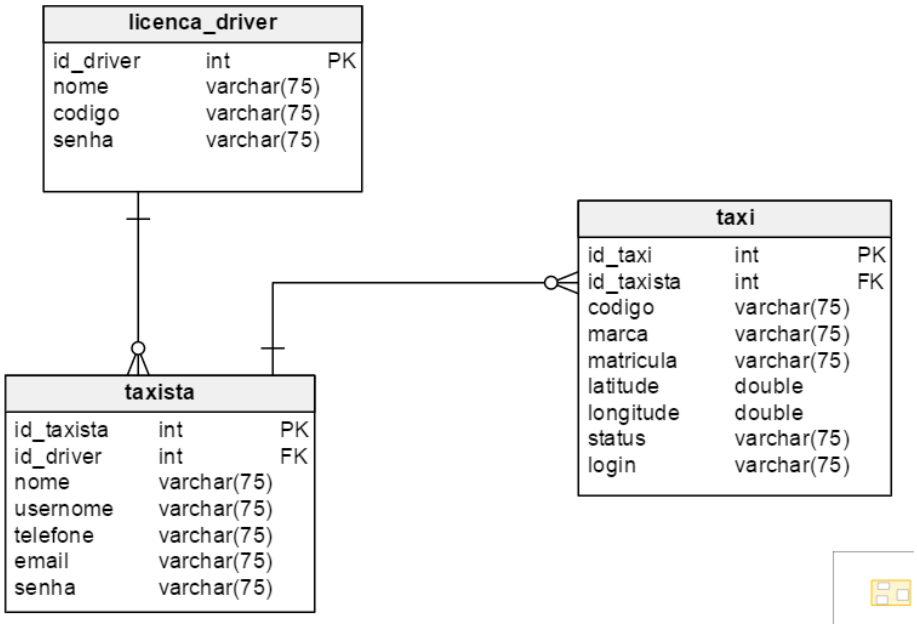

Figura 21: Base de dados

Fonte: Elaborado pelo autor

3.2.9. DICIONÁRIO DE DADOS

Em seguida segue-se o dicionário de dados para um melhor entendimento, que serão apresentados em tabelas todos os dados registrados no banco de dados e as suas descrições.

Na Tabela licenca_driver (TABELA 4), podemos encontrar os dados referente à licença de taxista. A Tabela taxista (TABELA 5) traz os dados do registro dos taxistas para o controle do acesso à sistema taxista e a tabela taxi (TABELA 6) encontramos os dados para a caracterização de um táxi, bem como os pontos em que um taxi encontra, neste caso são as coordenadas geográficas (latitude e longitude), armazenadas em formato *double*, pois são esses dados que nos permitirão obter a distância entre o cliente e o taxi bem como a apresentação da sua localização através da API Google Maps.

Tabela licenca_driver				
	Campo	**Tipo de dado**	**Tamanho**	**Observação**
PK	Id_driver	INT		Não nulo, AI
	nome	varchar	75	Não nulo
	codigo	varchar	75	Não nulo
	senha	varchar	75	Não nulo

Tabela 4: Licença de taxista

Fonte: Elaborado pelo autor

Tabela taxista				
	Campo	**Tipo de dado**	**Tamanho**	**Observação**
PK	id_taxista	INT		Não nulo, AI
FK	id_driver	INT		Não nulo
	nome	varchar	75	Não nulo
	usernome	varchar	75	Não nulo
	telefone	varchar	75	Não nulo
	email	varchar	75	Não nulo
	senha	varchar	75	Não nulo

Tabela 5: Dados do taxista

Elaborado pelo autor

Tabela taxi				
	Campo	**Tipo de dado**	**Tamanho**	**Observação**
PK	id_taxi	INT		Não nulo, AI
FK	id_taxista	INT		Não nulo
	codigo	varchar	75	Não nulo
	marca	varchar	75	Não nulo
	matricula	varchar	75	Não nulo
	latitude	double	75	Não nulo
	longitude	double	75	Não nulo
	status	varchar	75	Não nulo
	login	varchar	75	Não nulo

Tabela 6: Dados do taxi

Fonte: Elaborado pelo autor

4. DESENVOLVIMENTO

Neste capitulo fez-se uma abordagem sobre as ferramentas e tecnologia utilizadas no desenvolvimento do projeto e a demonstração alguns trechos de código e de algumas telas do protótipo abordada neste trabalho referente à interação do utilizador com o sistema.

4.1. Ferramentas e Tecnologias utilizadas

4.1.1. LINGUAGEM PROGRAMAÇÃO JAVA

Segundo SILVEIRA (2003), Java é uma linguagem de programação orientada a objeto desenvolvida na década de 90 por uma pequena equipa de programadores na empresa Sun Microsystems. Inicialmente elaborada para ser a linguagem-base de projetos de software para produtos eletrônicos, segundo PALMEIRA (2012) Java teve a sua grande expansão em 1995, visto que foi nessa altura que Sun viu uma oportunidade na Web, nessa época nas páginas não existia muita interatividade, apenas conteúdos estáticos eram exibidos. Então nesse ano a Sun anunciou o ambiente Java, sendo um absoluto sucesso, gerando uma aceitação aos browsers populares como o Netscape Navigator e padrões tridimensionais como o VRML (Virtual Reality Modeling Language - Linguagem de Modelagem para a Realidade Virtual).

Diferentemente das linguagens convencionais, que são compiladas para código nativo, a linguagem Java é compilada para um bytecode que por sua vez é executada por uma máquina virtual.

Característica

- Suporte à orientação a objetos;
- Portabilidade;
- Segurança;
- Linguagem Simples;
- Alta Performance;
- Dinamismo;

- Interpretada (o compilador pode executar os bytecodes do Java diretamente em qualquer máquina);
- Distribuído;
- Independente de plataforma;
- Tipada (detecta os tipos de variáveis quando declaradas);

Esta foi a linguagem utilizada no desenvolvimento da aplicação visto o Java é a linguagem base que a Google adotou para o desenvolvimento em Android.

4.1.2. ECLIPSE

Segundo a IBM (2013), O IDE Eclipse é uma ferramenta composta por elementos utilizados no desenvolvimento de software, tais como, editor de código, depurador, entre outros. O IDE Eclipse é mais frequentemente utilizado para o desenvolvimento com a linguagem Java, mas pode ser utilizado para desenvolver em outras linguagens mediante uso de plugin. Por exemplo, o plugin PDT é utilizado no eclipse para permitir o desenvolvimento com uso da linguagem PHP.

Segundo a GOOGLE (2012), o IDE Eclipse também possibilita fazer a integração com o Android SDK utilizando o ADT Plugin que é uma extensão desenvolvido para o Eclipse com um conjunto de ferramentas, utilizadas em projetos para o Android. O ADT Plugin oferece uma interface gráfica para construir interfaces de aplicações. Este plugin utiliza as bibliotecas do SDK, fazendo a adaptação entre o Android SDK e a IDE Eclipse.

Para este projeto o Eclipse foi o IDE utilizado para o desenvolvimento da aplicação.

4.1.3. ANDROID SDK

O SDK (Software Development Kit) da Android, inclui documentação, código e utilitários para que programadores consigam desenvolver as suas aplicações de acordo com um padrão de desenvolvimento para o sistema operativo em questão. A partir

dele pode-se emular softwares desenvolvidos para outras plataformas no Android e é a base para desenvolvedores criarem APP's para a plataforma do Google.
Pode ser obtido no site oficial da companhia e, com isso, pode-se contar com todos os recursos disponíveis nas novas versões do sistema operacional para a criação, adaptação e melhoria de aplicativos. Com ele, programadores podem desfrutar de todo o potencial oferecido pelo Google para o desenvolvimento de funcionalidades.

4.1.4. LINGUAGEM PHP

Segundo CASTELA (2010), o PHP (Hypertext Preprocessor) originalmente (Personal Home Page) é uma linguagem interpretada, livre, utilizada principalmente no desenvolvimento WEB. Sua Sintaxe lembra um pouco a sintaxe do C e do Perl e é uma linguagem bem fácil de aprender.

O PHP se diferencia de outros scripts porque ao invés de se escrever um monte de comandos para imprimir os HTML, é escrito um arquivo HTML com os códigos PHP embutidos entre o HTML delimitado por tags de início e fim.

O PHP é multiplataforma, podendo ser usado na maioria dos Sistemas Operacionais, OpenSource, e diferente de scripts como o JavaScript, ele roda no servidor, que aliás é suportado pela maioria dos servidores WEB que existem hoje no mercado como o Apache, IIS, PWS, etc. O cliente recebe apenas os resultados dos scripts, que são interpretados no servidor, não tendo acesso ao código. Muito simples de se aprender e trabalhar, o PHP atende desde os mais experientes desenvolvedores quanto os iniciantes na área.

As principais características do PHP são:

- Velocidade e robustez.
- Estruturado e orientação a objetos.
- Portabilidade - independência de plataforma.
- Tipagem dinâmica.
- Sintaxe similar a C/C++ e o Perl.

- Open-source.
- Server-side (O cliente manda o pedido e o servidor responde em página HTML).

Esta linguagem foi a escolhida para a implementação do webservice que será o responsável para o tratamento das atualizações e requisições do sistema.

A escolha deve-se ao factos de ser uma linguagem na qual eu tenho algum conhecimento e domino adquirido durante o curso de Informática de Gestão na Universidade do Mindelo, e sobre tudo ás caraterísticas da linguagem já descritas acima.

4.1.5. BASE DE DADOS MYSQL

Segundo ALECRIM (2008), o MySQL é um sistema de gerenciamento de banco de dados (SGBD), mais populares que existe e, por ser otimizado para aplicações Web, é amplamente utilizado na internet, utiliza a linguagem SQL (Linguagem de Consulta Estruturada, do inglês Structured Query Language) como interface.

É muito comum encontrar serviços de hospedagem de sites que oferecem o MySQL e a linguagem PHP, justamente porque ambos trabalham muito bem em conjunto.

Outro fator que ajuda na popularidade do MySQL é sua disponibilidade para praticamente qualquer sistema operacional, como Linux, FreeBSD (e outros sistemas baseados em Unix), Windows e Mac OS X. Além disso, o MySQL é um software livre (sob licença GPL), o que significa que qualquer um pode estudá-lo ou alterá-lo conforme a necessidade.

Características

- Portabilidade (suporta praticamente qualquer plataforma atual);
- Alta compatibilidade com linguagens como PHP, Java, Python, C#, Ruby e C/C++.
- Baixa exigência de processamento (em comparação como outros SGBD);

- Vários sistemas de armazenamento de dados (batabase engine), como MyISAM, MySQL Cluster, CSV, Merge, InnoDB, entre outros;
- Recursos como transactions (transações), conectividade segura, indexação de campos de texto, replicação, etc.
- Instruções em SQL, como indica o nome.

A sua escolha deve-se ao fato de ser um software livre, que permite uma fácil integração com a linguagem PHP, que é a linguagem escolhida para implementar o servidor, encontra-se disponível juntamente com o servidor Apache num único pacote, através do WampServer para ambiente Windows ou XampServer para ambiente Linux permitindo uma administração e configuração do sistema centralizada numa única maquina e por último por ser um sistema na qual eu tenho desenvolvido algum trabalhos acadêmicos durante o curso de Informática de Gestão na Universidade do Mindelo

4.1.6. WEBSERVICE RESTFUL

Segundo BEZERRA (2013), WebService são serviços/métodos que as aplicações disponibilizam para serem consumidos/executados por outras aplicações. Envolvem um produtor e um consumidor e são disponibilizados via Internet. São aplicações que, geralmente, executam em cima do protocolo HTTP, são sistemas distribuídas e seus componentes podem ser executados em dispositivos diversos tais como: desktops, smartphones e tabletes.

De acordo com MARTINS (2010), utilizar um WebService é uma das maneiras mais comuns de se integrar aplicações diferentes. Existem diferentes tipos de arquiteturas para webservices, e o RESTful é mais simples em comparação aos outros webservices, que geralmente utilizam a arquitetura SOAP (Simple Object Access Protocol, em português Protocolo Simples de Acesso a Objetos).

De acordo com SAUDATE (2013), REST é uma sigla que significa Representational State Transfer, é um modelo arquitetural concebido pelo doutor Roy Fielding (autor do protocolo HTTP) no ano de 2000, na sua dissertação de doutorado, na qual buscou

as melhores práticas nos estilos de arquiteturas existentes para compor um novo estilo que as reunissem em apenas um estilo, e tem como plataforma justamente as capacidades do protocolo, onde se destacam:

- Diferentes métodos (ou verbos) de comunicação (GET, POST, PUT, DELETE,

HEAD, OPTIONS);

- Utilização de headers HTTP (tanto padronizados quanto customizados);
- Definição de arquivos como recursos (ou seja, cada um com seu próprio endereço);
- Utilização de media types.

Segundo a NETBEANS, os Web Services RESTful são serviços construídos com o estilo de arquitetura RESTful. A construção de Web Services com a abordagem RESTful está surgindo como uma alternativa popular ao uso de tecnologias baseadas em SOAP para implantação de serviços na Internet, por ser mais leve e ter a capacidade de transmitir dados diretamente via HTTP.

É baseado em todas essas caraterísticas descritas acima que se optou por desenvolver um WebService baseado na arquitetura RESTful, como forma de comunicação da aplicação Taxista e a aplicação Cliente com a base de dados remota.

4.1.6.1. JSON

Segundo SAUDATE (2013, p.57), JSON (Javascript Object Notation) é uma linguagem de marcação criada por Douglas Crockford e descrito na RFC 4627, e serve como uma contrapartida a XML. Tem por principal motivação o tamanho reduzido em relação a XML, e acaba tendo uso mais propicio em cenários onde largura de banda (ou seja, quantidade de dados que pode ser transmitida em um determinado intervalo de tempo) e um recurso crítico.

Muitas linguagens hoje em dia dão suporte ao JSON, através das várias API's que tem sido desenvolvido para as várias linguagens, é meio que um novo método, substituto do antigo e conhecido XML.

Ele é muito usado para retornar dados vindos de um servidor utilizando requisições AJAX para atualizar dados em tempo real e tem sido muito utilizado em arquiteturas do tipo REST como formato de envio e recebimento de mensagens.

Exemplo de representação de um objeto em JSON

```
{
        usuario: {
            id: 123456,
            nome: "Maria Duarte",
            username: "mary123",
            email: "mary@xl.com"
        }
}
```

Exemplo de representação de um Array em JSON

```
{
    "notas": [
        {"nome": "Paulo", "nota1":14, "nota2":16, "nota3":15},
        {"nome": "Joana", "nota1":15, "nota2":19, "nota3":18}
    ]
}
```

Para o projeto em questão, o JSON foi utilizado como formato de dados para enviar e receber dados entre a aplicação e o WebService.

4.1.7. FÓRMULA HAVERSINE

A fórmula de Haversine é uma importante equação usada em navegação, fornecendo distâncias entre dois pontos de uma esfera a partir de suas latitudes e longitudes.

Segundo a formula dois pontos de uma esfera (de raio R) com latitudes $\varphi 1$ e $\varphi 2$, separação de latitude $\Delta\varphi = \varphi 1 - \varphi 2$, e separação de longitude $\Delta\lambda$, onde os ângulos são

em radianos, a distância d entre dois pontos (entre um círculo maior) da esfera) é relacionada as suas localizações pela fórmula: (a fórmula de Haversine)

$$\operatorname{haversin}\left(\frac{d}{R}\right) = \operatorname{haversin}(\Delta\phi) + \cos(\phi_1)\cos(\phi_2)\ \operatorname{haversin}(\Delta\lambda)$$

Onde h denota Haversine (d/R), dado acima. Podemos obter d tanto pela simples aplicação da Haversine inversa, quanto pelo uso da função arcos seno (inverso do seno).

$$d = R\ \operatorname{haversin}^{-1}(h) = 2R \arcsin\left(\sqrt{h}\right)$$

Esta fórmula é só uma aproximação quando aplicada à Terra, porque esta não é uma esfera perfeita: seu raio varia de 6356,78 km nos pólos até 6378,14 km no equador. Estas pequenas correções, na ordem de 0,1% (supondo R = 6367,45 km) são usadas em todo lugar, devido a leve forma elipsoide do nosso planeta.

4.1.8. GENYMOTION

Genymotion é um emulador do Android desenvolvido pela Genymobile, com um completo conjunto de sensores e características para interagir com um ambiente virtual.

É muito utilizado por desenvolvedores que precisam ou desejam testar as aplicações e as funcionalidades desta plataforma a partir de um sistema operacional Windows, Linux ou OSX usando uma VirtualBox.

O código de imagens virtuais é de código aberto, no entanto, o software que é executado no host (o "player") não é, mas é livre para usar. No futuro, Genymotion pretende continuar a ter versões livre para usar e com muitos recursos avançados, mas também haverá versões pagas, "principalmente para grandes empresas que necessitam de colaboração em Genymotion".

Características do Genymotion

- Permite baixar e executar imagens virtuais pré-configuradas: Android 4.1.1 – API nível 16 (com x86 support): Nexus 7 Jelly Bean, Nexus S Jelly Bean, Nexus One Jelly Bean, 10.1", WXGA Tablet Jelly Bean, 7.0" e WSVGA Tablet Jelly Bean, todos com ou sem Google Apps (Play Store, etc.);
- Rede: Ethernet (emula a conexão Wi-Fi);
- GPS (com coordenadas configuráveis) e widgets de emulação de bateria (com níveis de bateria configuráveis);
- Display: a aceleração de hardware OpenGL, múltiplas telas, exibição em tela cheia;
- Genymotion shell, que permite que você interaja com sua máquina virtual usando uma linha de comando;
- Suporte a ADB;
- Eclipse e Android Estúdio plug-ins;
- Suporta Linux, Windows e Mac.

Pré-requisitos

Sistema Operacional

- Microsoft Windows Vista, 7, 8/8.1 (32 ou 64 bits);
- Mac OS X 10.6 ou acima.
- Linux Ubuntu 12.04 (Precise Pangolin) ou acima.
- Linux Debian 7 (Wheezy).

Sistema

- Placa de vídeo atualizada e com suporte ao OpenGL 2.0;
- Suporte a VT-x ou AMD-V (habilitado nas configurações do BIOS);
- Pelo menos 2GB de memória RAM;
- Resolução de tela maior que 1024 X 768 pixels;

- Pelo menos 100MB de espaço livre no disco rígido;
- Observação: Um mínimo de 2GB de espaço livre é necessário para rodar a máquina virtual.

Máquina Virtual

Para executar os dispositivos virtuais do Genymotion é necessário instalar o Oracle VM VirtualBox (versão 4.1 ou acima). O site oficial do Genymotion recomenda por questões de performance que seja usada a versão 4.3.12.

Para o projeto em questão, o Genymotion foi utilizado como emulador para testar as funcionalidades da aplicação.

4.1.9. GOOGLE MAPS V2

A integração do Google Maps com uma aplicação Android requer uma chave de autenticação. Esta chave é necessária para que seja possível acessar o banco de dados da Google e obter as informações dos mapas.

Desde 3 de dezembro de 2012 a v1 da Google Maps API foi descontinuada. As chaves obtidas para a primeira versão da API não podem ser utilizadas nesta nova versão. Desta forma, é necessário gerar uma chave específica para esta versão da API.

Para esta nova versão podem ser geradas chaves do tipo debug e do tipo release. As chaves do tipo debug são designadas para aplicações em testes, que não serão publicadas na loja da Google. As chaves do tipo release são específicas para disponibilizar aplicações na loja. Para as aplicações desenvolvidas, foram utilizadas chaves do tipo debug.

O método para obter as chaves de testes requer alguns passos. Ao executar a aplicação Android através do Eclipse, é gerado um arquivo debug.keystore, utilizado para

extrair através da ferramenta keytool do Java o "SHA-1 fingerprint". É necessário também uma conta Google para registrar o projeto na Google APIs Console, informar a SHA-1 fingerprint obtida e obter a chave de utilização do Google Maps. Esta chave deve ser inserida no AndroidManifest.

4.2. Protótipo

Segundo NEPOMUCENO (2012), um protótipo é uma versão inicial de um sistema de software, que é utilizada para mostrar conceitos, experimentar opções de projeto e, em geral, para conhecer mais sobre os problemas e suas possíveis soluções. O desenvolvimento rápido de um protótipo é essencial para que os custos sejam controlados e os utilizadores possam fazer experiências com o protótipo no início do processo de software.

Ainda segundo o mesmo, um protótipo de software apoia duas atividades do processo de engenharia de requisitos:

Levantamento de requisitos - Os protótipos de sistema permitem que os utilizadores realizem experiências para ver como o sistema apoia seu trabalho. Eles obtêm novas ideias para os requisitos e podem identificar pontos positivos e negativos do software. Eles podem, então, propor novos requisitos de sistema.

Validação de requisitos - O protótipo pode revelar erros e omissões nos requisitos propostos. Uma função descrita em uma especificação pode parecer útil e bem-definida. Contudo, quando essa função é utilizada com outras, os utilizadores muitas vezes acham que sua visão inicial era incorreta e incompleta. A especificação de sistema pode então ser modificada para refletir sua compreensão alterada dos requisitos.

Na maioria dos projetos, o primeiro sistema construído dificilmente será usável. Ele pode ser muito lento, muito grande, desajeitado em uso, ou todos os três. A questão

administrativa, não é se deve construir um sistema-piloto e jogá-lo fora. Isso será feito. A única questão é se deve planejar antecipadamente a construção de algo que se vai jogar fora ou prometer entregar isso aos clientes.

O protótipo pode ser oferecido ao cliente em diferentes formas:

- Protótipo em papel;
- Modelo executável em PC retratando a interface homem-máquina; capacitando o cliente a compreender a forma de interação com o software;
- Protótipo de trabalho que implemente um subconjunto dos requisitos indicados;
- Programa existente (pacote) que permita representar todas ou partes das funções desejadas para o software a construir.

Vantagens da prototipação:

- Modelo de desenvolvimento interessante para alguns sistemas de grande porte que representem um certo grau de dificuldade para exprimir rigorosamente os requisitos;
- É possível demonstrar a realizabilidade através da construção de um protótipo do sistema;
- É possível obter uma versão, mesmo simplificada do que será o sistema, com um pequeno investimento inicial;
- A experiência adquirida no desenvolvimento do protótipo vai ser de extrema utilidade nas etapas posteriores do desenvolvimento do sistema real, permitindo reduzir o seu custo e resultando num sistema melhor concebido.

Problemas da prototipação:

- Quando informamos que o produto precisa ser reconstruído, o cliente exige que alguns acertos sejam aplicados para tornar o protótipo um produto; muito frequentemente, a gerência de desenvolvimento de software cede;

- O desenvolvedor muitas vezes faz concessões de implementação a fim de colocar um protótipo em funcionamento rapidamente. Depois de algum tempo, o desenvolvedor pode familiarizar-se com essas opções e esquecer-se de todas as razões pelas quais elas são inadequadas - a opção menos ideal se tornou então parte integrante do sistema;

4.2.1. CONFIGURAÇÃO DA APLICAÇÃO

Como referido no capitulo dois em que abordou-se sobre a plataforma Android falou-se da existência de um arquivo chamado AndroidManifest.xml, que é obrigatório e é nele que são feitas as configurações de todos os recursos utilizados pela aplicação (activities, componentes gráficos, layouts, permissões, imagens, etc.).

As figuras a seguir apresentam o conteúdo comentado do arquivo AndroidManifest.xml da aplicação Cliente e da aplicação Taxista.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.umtaxi"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="10"
        android:targetSdkVersion="21" />

<!-- Permission for aplication use -->
    <permission
        android:name="com.umtaxi.permission.MAPS_RECEIVE"
        android:protectionLevel="signature" />

    <uses-permission android:name="com.umtaxi.permission.MAPS_RECEIVE" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.CALL_PHONE"/>
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>
    <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Figura 22: Configurações das permissões de uso da aplicação cliente

```
<!-- Aplication Activities-->
        <activity
    android:name="com.umtaxi.InicioActivity"
    android:label="@string/app_name" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

<activity
    android:name="com.umtaxi.ListaTaxisActivity"
     android:label="@string/app_name" > </activity>

<activity
    android:name="com.umtaxi.Taxis_proximosActivity"
      > </activity>

<activity
    android:name="com.umtaxi.Formulario"
     ></activity>

<activity
```

Figura 23: Configuração das principais Activities da aplicação Cliente

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.umtaxicondutor"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="9"
        android:targetSdkVersion="21" />


    <!-- Permission for aplication use -->
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Figura 24: Configurações das permissões de uso da aplicação Taxista

```
        <!-- Aplication Activities-->
        <activity
            android:name="com.umtaxicondutor.LoginActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity
            android:name="com.umtaxicondutor.Registrar"></activity>


        <activity
            android:name="com.umtaxicondutor.DadosTaxista"></activity>

        <activity
            android:name="com.umtaxicondutor.Licenca_taxista"></activity>

        <activity
            android:name="com.umtaxicondutor.Selecionar_taxi"></activity>

        <activity
            android:name="com.umtaxicondutor.RegistrarTaxi"></activity>


        <activity
            android:name="com.umtaxicondutor.LocationActivity"></activity>
    </application>
```

Figura 25: Configuração das principais Activities da aplicação Taxista

4.2.2. GOOGLE MAPS KEY

A integração do Google Maps com uma aplicação Android requer uma chave de API. A chave é utilizada para que os servidores do Google reconheçam a aplicação e permitam o acesso aos dados dos mapas. Desta maneira, a Google previne que uma aplicação acesse os servidores de maneira descontrolada deixando-os lentos e possivelmente indisponíveis para outras aplicações como já foi explicada no quarto capitulo em que falou-se sobre as Google Maps.

Após obter essa chave ela deve ser inserida no AndroidManifest juntamente com a sua referida versão.

A figura 26 apresenta como é que essa chave é introduzida na aplicação.

```
<!-- Required OpenGL ES 2.0. for Maps V2 -->
<uses-feature
    android:glEsVersion="0x00020000"
    android:required="true" />

<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >

    <!-- Goolge API Key -->
     <meta-data
        android:name="com.google.android.maps.v2.API_KEY"
        android:value="CHAVE OBTIDA DO GOOGLE" />
```

Figura 26: Chave da Google Maps e a sua versão

Depois de configurada a chave do API Google Maps para que o utilizador possa utilizar os recursos dessa mesma API deve-se adicionar algumas permissões como pode-se constatar na figura 27:

```
<permission
    android:name="com.umtaxi.permission.MAPS_RECEIVE"
    android:protectionLevel="signature" />
<uses-permission android:name="com.umtaxi.permission.MAPS_RECEIVE" />
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Figura 27: Permissões para uso da API Google Maps

4.2.3. OBTENÇÃO DA LOCALIZAÇÃO DO APARELHO

Para obter a localização do utilizador é utilizada GPS do dispositivo em conjunto com o provedor WiFi. Para tal é utilizada os serviços de localização que o Android disponibliza e que podem ser acessadas através do pacote android.location.

Criou-se uma classe denominada de GPSTracker que utiliza as seguinte classe desse pacote:

- **LocationManager-** O componente central deste framework de localização é o serviço que fornece a API responsável por determinar a localização geográfica. Assim como outros serviços de sistema, não é possível instancia-lo diretamente. Para utiliza-lo deverá ser a feita uma requisição da sua instância pelo sistema da seguinte forma:

```
LocationManager locationManager =
(LocationManager) this.getSystemService(Context.LOCATION_SERVICE);
```

Figura 28: Instancia da classe LocatioManager

- **LocationProvider -** permite especificar o provedor de localização que fornece as informações acerca da posição atual do dispositivo.
- **Location -** permite solicitar a localização do dispositivo. A localização do dispositivo pode conter informações de latitude, longitude, altitude, entre outros.

Na figura 29 pode-se verificar um extrato do código fonte de como foi utilizado as referidas classes no método getLocation.

```
public Location getLocation() {
    try {
        locationManager = (LocationManager) mContext
                .getSystemService(LOCATION_SERVICE);

        // getting GPS status
        isGPSEnabled = locationManager
                .isProviderEnabled(LocationManager.GPS_PROVIDER);

        // getting network status
        isNetworkEnabled = locationManager
                .isProviderEnabled(LocationManager.NETWORK_PROVIDER);

        if (!isGPSEnabled && !isNetworkEnabled) {
            // no network provider is enabled
        } else {
            this.canGetLocation = true;
            if (isNetworkEnabled) {
                locationManager.requestLocationUpdates(
                        LocationManager.NETWORK_PROVIDER,
                        MIN_TIME_BW_UPDATES,
                        MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
                Log.d("Network", "Network");
                if (locationManager != null) {
                    location = locationManager
                            .getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
                    if (location != null) {
                        latitude = location.getLatitude();
                        longitude = location.getLongitude();
                    }
                }
            }
            // if GPS Enabled get lat/long using GPS Services
            if (isGPSEnabled) {
                if (location == null) {
```

Figura 29: Método getLocation

4.2.4. ACESSO AO WEBSERVICE

Para a realização da leitura e da gravação de dados em banco de dados através da aplicação Android foi necessária a utilização de um webservice, utilizando os protocolos HTTP POST e GET e o formato JSON para a troca de dados.

A figura 30 apresenta um extrato do código fonte de como foi implementada os métodos HTTP POST e GET:

```
// function get json from url
// by making HTTP POST or GET mehtod
public JSONObject makeHttpRequest(String url, String method,
        List<NameValuePair> params) {

    // Making HTTP request
    try {

        // check for request method
        if(method == "POST"){
            // request method is POST
            // defaultHttpClient
            DefaultHttpClient httpClient = new DefaultHttpClient();
            HttpPost httpPost = new HttpPost(url);
            httpPost.setEntity(new UrlEncodedFormEntity(params));

            HttpResponse httpResponse = httpClient.execute(httpPost);
            HttpEntity httpEntity = httpResponse.getEntity();
            is = httpEntity.getContent();

        }else if(method == "GET"){
            // request method is GET
            DefaultHttpClient httpClient = new DefaultHttpClient();
            String paramString = URLEncodedUtils.format(params, "utf-8");
            url += "?" + paramString;
            HttpGet httpGet = new HttpGet(url);

            HttpResponse httpResponse = httpClient.execute(httpGet);
            HttpEntity httpEntity = httpResponse.getEntity();
            is = httpEntity.getContent();
        }
```

Figura 30: Método POST e GET

4.2.5. OBTENÇÃO DOS DADOS DOS TAXIS PRÓXIMOS

Após fazer uma requisição ao servidor este retorna os dados em formato JSON como pode-se constatar na seguinte figura na qual foi feita uma requisição para obter a lista dos táxis próximos e as suas respectivas informações:

{"veiculos":[{"id":"1","nome":"Frederico Soares","usernome":"fred","email":"soares@gmail.com","telefone":"9854717","matricula":"SV-34-
ft","marca":"BMW","status":"off","latitude":"16.8972909","longitude":"-24.9863751"},{"id":"2","nome":"Mario
Oliveira","usernome":"Mario","email":"mario@gmail.com","telefone":"9874752","matricula":"ST-56-GE","marca":"TOYOTA","status":"on","latitude":"16.897736","longitude":"-24.986878"},
{"id":"3","nome":"Antonio Duarte","usernome":"Antonio","email":"antonio@gmail.com","telefone":"9638746","matricula":"SV-65-
DF","marca":"FORD","status":"on","latitude":"16.897797","longitude":"-24.986845"},{"id":"4","nome":"Paulo Lopes","usernome":"Paulo","email":"paulo@gmail.com","telefone":"9685475","matricula":"SV-
68-FG","marca":"OPEL","status":"on","latitude":"16.897778","longitude":"-24.986889"},{"id":"5","nome":"Pedro
Santos","usernome":"Pedro","email":"pedro@hotmail.com","telefone":"9784561","matricula":"ST-23-SD","marca":"OPEL","status":"on","latitude":"16.897745","longitude":"-24.986845"},
{"id":"6","nome":"Olavo Silva","usernome":"Olavo","email":"olavo@gmail.com","telefone":"9874512","matricula":"SV-87-
SH","marca":"TOYOTA","status":"on","latitude":"16.897759","longitude":"-24.986864"},{"id":"7","nome":"Joao Gomes","usernome":"Jo","email":"jo@hotmail.com","telefone":"9652382","matricula":"SV-
46-KT","marca":"FORD","status":"off","latitude":"16.8973937","longitude":"-24.9864209"},{"id":"8","nome":"Sandro
Oliveira","usernome":"San","email":"san@gmail.com","telefone":"9845213","matricula":"SV-78-TF","marca":"OPEL","status":"off","latitude":"16.8971382","longitude":"-24.9866083"},
{"id":"9","nome":"Mateus Silva","usernome":"mat","email":"mat@gmail.com","telefone":"9855213","matricula":"SV-56-
GT","marca":"OPEL","status":"off","latitude":"16.8891332","longitude":"-24.9888638"},{"id":"10","nome":"Elvis Araujo","usernome":"vis","email":"vis@gmail.com","telefone":"9856321","matricula":"SV-
98-DR","marca":"OPEL","status":"on","latitude":"16.8963419","longitude":"-24.9887588"},{"id":"11","nome":"Pedro","usernome":"ped","email":"ped@gmail.com","telefone":"9854216","matricula":"SV-57-
FG","marca":"OPEL","status":"off","latitude":"16.8984064","longitude":"-24.987742"}],"success":1}

Figura 31: Dados dos taxis em formato JSON

O aplicativo ao receber esses dados no formato JSON, irá processa-los e transformá-los em objetos utilizáveis pelo aplicativo. Para processar esses dados em formato JSON utilizou-se uma biblioteca disponível no Android especifica para essa finalidade denominada de “org.json”.

O estrato de código a seguir demostra parte de como é transformado os dados JSON recebido da consulta dos táxis próximos.

```
protected String doInBackground(String... args) {
    // Building Parameters
    List<NameValuePair> params = new ArrayList<NameValuePair>();
    // getting JSON string from URL
    JSONObject json = jParser.makeHttpRequest(url_all_veiculos, "GET", params);

    // Check your log cat for JSON reponse
    Log.d("Todos os Veiculos: ", json.toString());

    try {
        // Checking for SUCCESS TAG
        int success = json.getInt(TAG_SUCCESS);

        if (success == 1) {
            // veiculos encontrados
            // Getting Array de veiculos
            veiculos = json.getJSONArray(TAG_VEICULOS);

            dis = new Distancia();

            Log.d("minha latitude: ", String.valueOf(latitude));
            Log.d("meu longitude: ",String.valueOf(longitude));

            int res;

            // looping through todos os veiculos
            for (int i = 0; i < veiculos.length(); i++) {
                JSONObject c = veiculos.getJSONObject(i);
                //calcular distancia
                res = (int) dis.calcularDistancia(latitude, longitude, c.getDouble("latitude"), c.getDouble("longitude"));

                Veiculo vei = new Veiculo(c.getLong(TAG_PID),c.getString(TAG_NAME),
                        c.getString("username"),c.getString("email"),c.getString("telefone"),
                        c.getString("matricula"),c.getString("marca"),c.getString("status"),
                        c.getDouble("latitude"),c.getDouble("longitude"),res);
                // adicionar os objectos veiculos no ArrayList
                veiculosList.add(vei);

            }
```

Figura 32: Extrato do código da transformação dos dados JSON

4.2.6. OBTENDO A ROTA

Para obter a rota entre dois pontos foi criada um método denominada de getRoute que recebe como parâmetro a localização do cliente e do taxista e irá requisitar os dados da rota aos serviços da Google através de uma URL disponibilizada pela mesma em que essa resposta será em formato JSON contendo todos os dados referente à rota.

No código a seguir pode-se conferir a implementação do método getRoute.

```
public void getRoute(final LatLng origin, final LatLng destination){
    new Thread(){
        public void run(){
            /*String url= "http://maps.googleapis.com/maps/api/directions/json?origin="
                    + origin+"&destination="
                    + destination+"&sensor=false";*/
            String url= "http://maps.googleapis.com/maps/api/directions/json?origin="
                    + origin.latitude+","+origin.longitude+"&destination="
                    + destination.latitude+","+destination.longitude+"&sensor=false";


            HttpResponse response;
            HttpGet request;
            AndroidHttpClient client = AndroidHttpClient.newInstance("route");

            request = new HttpGet(url);
            try {
                response = client.execute(request);
                final String answer = EntityUtils.toString(response.getEntity());

                runOnUiThread(new Runnable(){
                    public void run(){
                        try {
                            //Log.i("Script", answer);
                            list = buildJSONRoute(answer);
                            drawRoute();
                        }
                        catch(JSONException e) {
                            e.printStackTrace();
                        }
                    }
                });

            }
            catch(IOException e) {
                e.printStackTrace();
```

Figura 33: Método para obter a rota entre dois pontos

4.2.7. FUNCIONAMENTO

O funcionamento das aplicações é apresentado com base nos casos de uso descritos, sendo apresentadas telas das aplicações em execução e descrição dos passos para melhor entendimento.

As interfaces do protótipo são simples e foram desenvolvidas com o objetivo de facilitar a utilização para os utilizadores não habituados a dispositivos móveis. As imagens apresentadas foram registradas, realizando uma solicitação de serviço do endereço da Universidade do Mindelo.

4.2.7.1. APLICAÇÃO CLIENTE

A aplicação do cliente possui basicamente três telas. Ao iniciar a aplicação (figura 34) o cliente é orientado através do botão "Lista Taxis". Neste caso, é necessária a utilização do GPS para obter a posição do dispositivo e de internet para obter o endereço do utilizador. Caso o cliente não esteja com estes serviços habilitados, é apresentada opção para ativar os mesmos. Ao confirmar a ativação, o cliente já pode obter a lista dos táxis (figura 35) disponíveis após clicar no botão "Lista Taxis"

Figura 34: Tela inicial APP Cliente

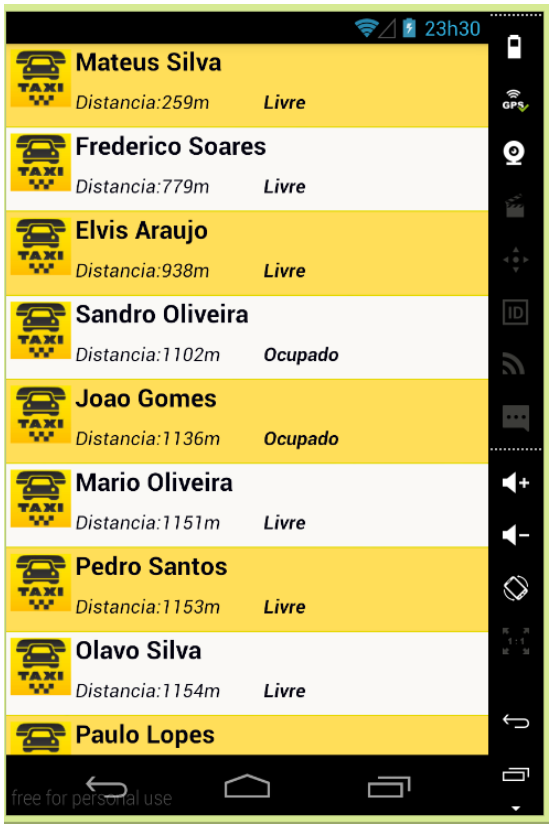

Figura 35: Tela lista taxis

Após abrir a tela com a lista dos taxis podemos observar os nomes dos taxistas a distância que encontram em relação ao cliente bem como os seus estados no momento. Ao fazer o clique longo no táxi pretendido será apresentado um menu (figura 36) com as opções de ligar caso o cliente queira ligar o referido taxista (figura 37) e de enviar SMS caso o cliente queira envia-lo uma mensagem escrita como mostra a figura 38.

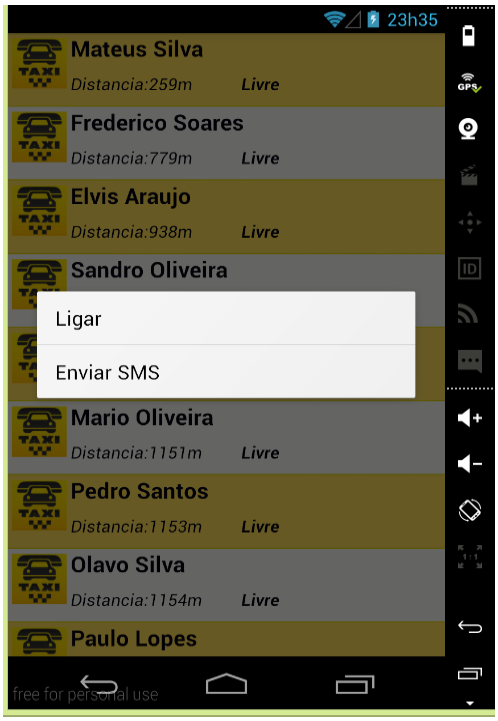

Figura 36: Menu requisição taxis

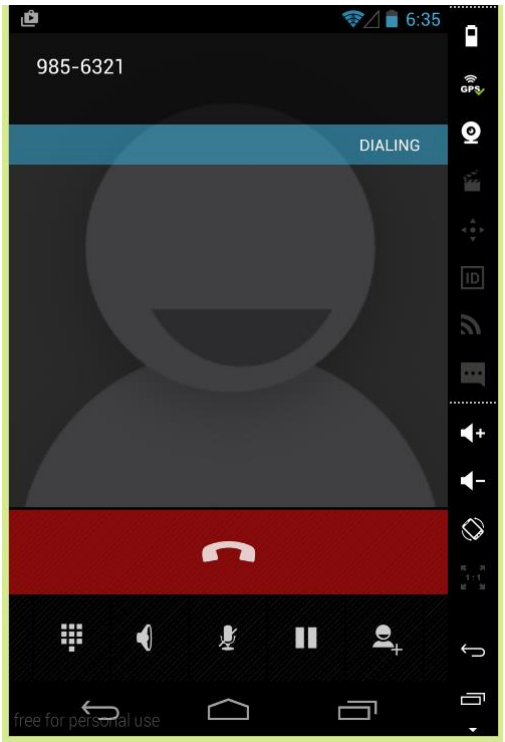

Figura 37: Chamada Telefonica APP Cliente

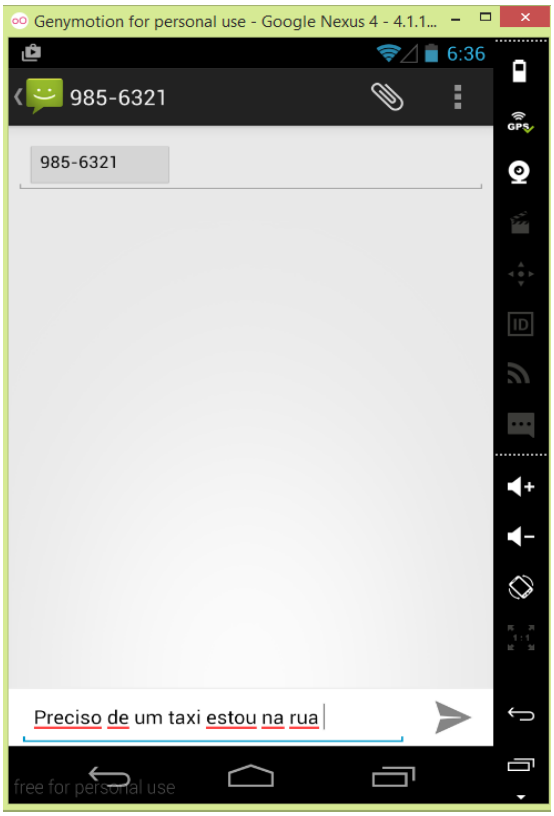

Figura 38: Enviar SMS APP Cliente

Caso o cliente fizer um simples clique será aberta uma nova tela (figura 39), apresentando os dados do taxista a sua localização no mapa bem como a do cliente e a respectiva rota, ainda terá dois botões coso queira também ligar ou enviar um SMS.

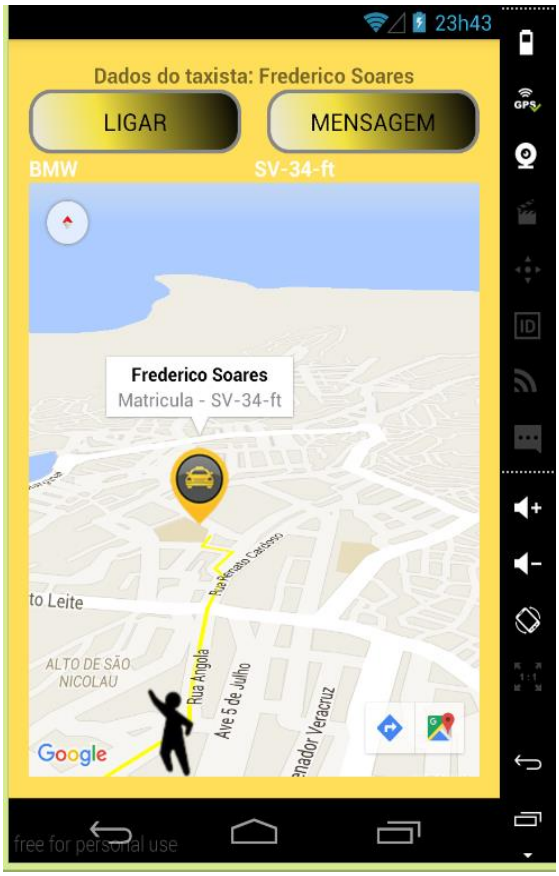

Figura 39: Dados do Taxista

4.2.7.2. APLICAÇÃO TAXISTA

Ao abrir o aplicativo devem ser informados o nome do utilizador e a senha, conforme demonstrado na figura 40 caso o taxista encontra-se registrado caso contrário é informado que deve-se registrar clicando na opção para registra-se. Para que essas e outras operações sejam efetuadas é necessária que o dispositivo esteja com uma ligação à internet ativa para poder comunicar com o WebService.

Figura 40: Tela inicial APP Taxista

Caso o utilizador clicar em registrar-se será aberta a tela de licença taxista como mostra a figura 41 onde deve-se informar os seus dados relativamente à licença de taxista. Após confirmado os dados será agora exibida a tela de registro onde este deve informar os seus dados pessoais como mostra a figura 42.

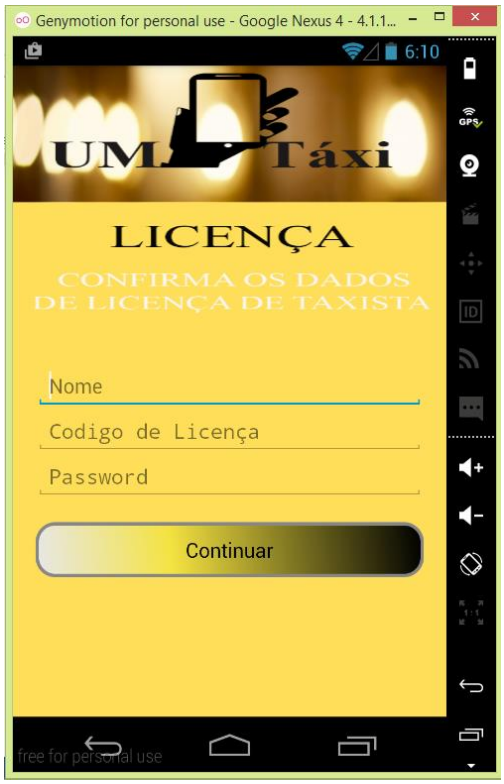

Figura 41: Tela Licença Taxista

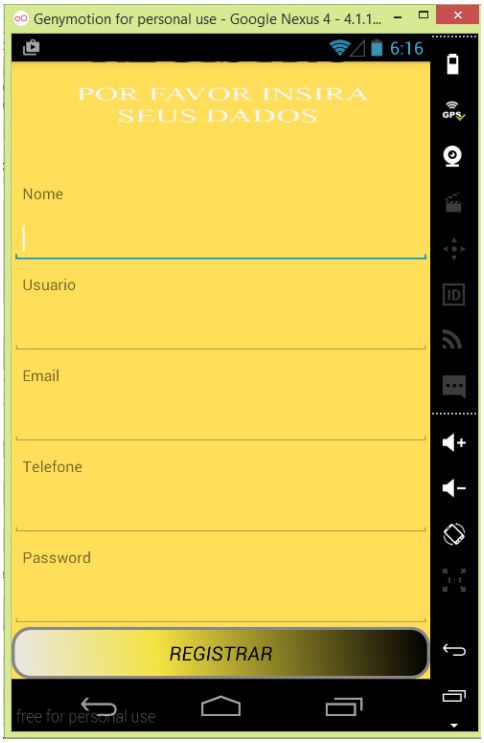

Figura 42: Tela registro de taxista

Depois de efetuado o registro o utilizador já poderá efetuar o login, após efetuar o login será apresentado a tela contendo os seus dados caso queira atualiza-los caso contrário poderá prosseguir clicando no botão continuar como demostra a figura 43.

Figura 43: Dados do taxista

Após clicar em continuar será exibida uma tela onde é pedido que indica as informações do taxi que pretende conduzir, caso o seu táxi não esteja registrado é pedido que registe o seu táxi como mostra a figura 44, onde é exibida a tela de registro do táxi como mostra a figura 45.

Figura 44: Dados do táxi

Figura 45: Tela registro do taxi

Após concluir o registro do táxi o utilizador já poderá entrar no sistema onde é exibido a seguinte tela como mostra a figura 46.

Figura 46: Tela Interação

Nessa tela o utilizador terá as opções de iniciar o sistema através do botão "Iniciar Atualização" que irá enviar as coordenadas geográficas para o servidor remoto a cada cinco segundos e a cada um metro percorrido, ainda terá mais dois botões um para alterar o seu estado e outra para sair do sistema.

5. CONCLUSÃO

5.1. Analise do trabalho desenvolvido

Este projeto tentou apresentar/implementar um aplicativo para dispositivos móveis, que utilizam a plataforma Android, e visa auxiliar os seus utilizadores a buscar e chamar os serviços de táxis mais próximos às suas localizações geográficas através de uma chamada telefônica ou de uma mensagem escrita. Para tal, foram reunidas as tecnologias Android, GPS e Google Maps, para criar dois aplicativos, um para uso dos clientes e outro para uso dos taxistas, que torne esta tarefa mais fácil e eficiente.

Após o desenvolvimento, iniciou-se o processo de avaliação dos aplicativos com simulação de resultados. Com base nos resultados da etapa de avaliação dos aplicativos, verificou-se que este trabalho atingiu seu objetivo em desenvolver um aplicativo para dispositivos móveis utilizando a plataforma Android para utilizadores dos serviços de táxi, de modo que este serviço prestado pelo sistema desenvolvido torne este processo mais ágil e eficiente aos seus utilizadores com destaque para a facilidade e simplicidade das ferramentas desenvolvidas.

5.2. Trabalhos futuros

Como continuação deste projeto, fica em aberto o estudo e a expansão do aplicativo para outras plataformas mobile nomeadamente a iOS e a Windows Phone.

A utilização de notificações em que o cliente envia uma solicitação ao taxista cabendo a este aceitar ou não.

Permitir que o utilizador possa enviar suas coordenadas geográficas ao taxista e este possa ver as referidas rotas para chegar até ao cliente.

Possibilitar que o cliente possa acompanhar o deslocamento do seu taxista em tempo real.

6. REFERÊNCIAS BIBLIOGRÁFICAS

BERNARDI, J.V.E. & LANDIM, P.M.B. "**Aplicação do Sistema de Posicionamento Global (GPS) na coleta de dados.** ". Disponivel em: http://www.academia.edu/1202158/Aplica%C3%A7%C3%A3o_do_Sistema_de_Posicionamento_Global_GPS_na_coleta_de_dados, Acesso em: 31-10-2015

BEZERRA, Luana Mayara F. S. **Desenvolvendo Web Services RESTFUL utilizando a API JAX-RS 2.0 e Jersey**. Disponível em: http://www.devmedia.com.br/desenvolvendo-web-services-restful-utilizando-a-api-jax-rs-2-0-e-jersey/27141, Acesso em: 15-04-2015

CAFÉ, Almeida Adriel. **Desenvolvimento de Cross-Platform Mobile Apps Utilizando o Titanium Mobile**. Disponível em: http://adrielcafe.com/artigos/74-tcc-desenvolvimento-de-cross-platform-mobile-apps-utilizando-o-titanium-mobile, Acesso em 05-10-2015

CASTELA, Rodrigo Tenedini. **Introdução a Linguagem PHP**. Disponível em: http://www.dotsharp.com.br/programacao/php/introducao-a-linguagem-php.html, Acesso em 21-06-2015

CHEDE, Cezar. **Desenvolvimento de apps – Parte 2: híbrido, nativo ou web**. Disponível em: https://www.ibm.com/developerworks/community/blogs/ctaurion/entry/desenvolvimento_de_apps-parte_2_hibrido_nativo_ou_web?lang=en, Acesso em 04-10-2015

DA SILVA, Francisco Mendes Citeli. **Entendendo o ciclo de vida de uma aplicação Android**. Disponível em: http://www.devmedia.com.br/entendendo-o-ciclo-de-vida-de-uma-aplicacao-android/22922, Acesso em: 14-03-2015

FIGUEIREDO, C. M. S. e NAKAMURA, E. (2003). **Computação móvel: Novas oportunidades e novos desafios.**

FITZ, P.R. "**Cartografia básica**". São Paulo: Oficina de textos, 2008.

GADELHA, Reginaldo. **Desenvolvendo para Android: Arquitetura Android**. Disponível em: http://www.tiselvagem.com.br/geral/desenvolvendo-para-android-arquitetura-android/, Acesso em 12-03-2015

GOOGLE. **Activities**. Disponível em: http://developer.android.com/guide/components/activities.html, Acesso em: 13-03-2015

IBM. **Intodução á plataforma Eclipse**. Disponível em: https://www.ibm.com/developerworks/br/library/os-eclipse-platform/, Acesso em: 18-06-2015

LECHETA, R. R. **Google Android: Aprenda a criar aplicações para dispositivos móveis com o Android SDK.** Novatec, 2ª edição, 2010.

MATEUS, G.R e LOUREIRO, A.F. "**Introdução a Computação Móvel**". Segunda edição, 1998, Minas Gerais.

MARTINS, Marcelo. **Criando um WebService Restful em Java**. Disponivel em: http://www.k19.com.br/artigos/criando-um-webservice-restful-em-java/, Acesso em: 15-04-2015

MONICO, J.F.G. "**Posicionamento pelo NAVSTAR-GPS**: Descrição, fundamentos e aplicações.". São Paulo: Editora UNESP, 2000

NEPOMUCENO, D. "**Modelo Incremental, Espiral e de Prototipação**". Disponível em: http://engenhariadesoftwareuesb.blogspot.com/2012/12/blog-post.html, Acesso em: 10-10-2015

NASA. **Global Positioning System History**. Disponível em: http://www.nasa.gov/directorates/heo/scan/communications/policy/GPS_History.html, Acesso em: 01-11-2015

NETBEANS. **Introdução aos Web Services RESTful**. Disponível em: https://netbeans.org/kb/docs/websvc/rest_pt_BR.html, Acesso em: 12-04-2015

NUNES, M e O'Neill, H (2004). **"Fundamental de UML"**.

PALMEIRA, Thiago Vinícius Varallo. **Java: história a principais conceitos**. Disponível em: http://www.devmedia.com.br/java-historia-e-principais-conceitos/25178, Acesso em: 12-04-2015

SAUDATE, ALEXANDRE (2013). **REST: construa API's inteligentes de maneira simples**

SAUDATE, ALEXANDRE (2013). **SOA aplicado: Integrando com Web Services e além**

SILVA, Alberto M. R. e VIDEIRA, Carlos A. E. (2001). **UML, Metodologias e Ferramentas CASE**

SILVEIRA, I.F**. Linguagem Java**. Disponível em: http://www.infowester.com/lingjava.php, Acesso em: 09-04-2015

TOSIN, Carlos. **Conhecendo o Android**. Disponível em: http://www.softblue.com.br/blog/home/postid/11/CONHECENDO+O+ANDROID, Acesso em: 12-03-2015

TAVARES, Alexandre. **iOS, o sistema móvel da Apple**. Disponível em: http://smartmundo.com/ios-sistema-movel-apple, Acesso em: 07-10-2015